PUB-3033

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Materials & Molecular
Research Division

SPEC-DOC: A USER'S GUIDE TO SPECTROMETER SOFTWARE

S. Sinton, J.R. Garbow, J.L. Ackerman, G. Drobny,
D.J. Ruben, and A. Pines

May 1983

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

Pub-3033

"SPEC-DOC: A User's Guide to Spectrometer Software"

S. Sinton, J. R. Garbow, J. L. Ackerman, G. Drobny,

D. J. Ruben and A. Pines

Lawrence Berkeley Laboratory, University of California, Berkeley, CA

This work was supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Materials Science Division of the U. S. Department of Energy under Contract Number DE-AC03-76SF00098. The authors wish to acknowledge the help of Debbie Garbow and Dione Carmichael in the preparation of this manuscript.

TABLE OF CONTENTS

I. INTRODUCTION

SPEC is the name of the operating system designed to control the NMR spectrometers in lab. SPEC is actually one large program which handles many functions necessary to control each spectrometer. The program handles all I/O with peripheral devices such as the console ("terminal" or "CRT"). The program carries out its operations by accepting commands which each invoke specific subroutines to perform their function. There are a total of 60 commands in SPEC, each carrying out a different function. Because so many commands make SPEC a very large program, not all of the program is core resident. Rather, each command calls in an overlay handler which loads into memory the appropriate overlay from the disk and begins execution of the command. Thus SPEC is an independent disk based operating system.

The commands in SPEC are capable of operating the microprocessor based pulse programmer, starting and acquiring data from the spectrometer data acquisition system, storing data on disk and manipulating it mathematically, displaying and plotting data. All arithmetic operations within SPEC are performed on integers. Since the DATA GENERAL computers are 16 bit machines operating in two's complement mode, the integer range is $\pm$32767. Many of the mathematical operations of SPEC are done in double precision integer mode with the final result always scaled to the above range. For many of the commands, integer overflow is detected and reported as an error message. Overflowed points are set to $\pm$32767.

SPEC accepts command input from the console or reads a string of commands previously entered on the disk. The later command structure is called a macro. Macros may be nested and may have constants passed to them at execution time, thus allowing for a powerful supercommand structure. Both forms of commands are discussed in the next section.

SPEC is designed to run on a DATA GENERAL computer with 32K words of memory and a 10Mbyte hard disk system. There are minor differences between the various computers in lab which are discussed in one of the appendices. The disk systems consist of two, 5Mbyte platters, one fixed and one in a removable pack. While running SPEC, all disk I/O is with the fixed disk. All pulse programs, data records, phase box records, etc. reside on the fixed disk. Thus each disk drive is specific to one spectrometer. When running under RDOS (the DG operating system) these disk files may also be accessed.

All the source listings of SPEC subroutines, the SPEC program library, and all supporting programs reside on one of the removable packs (labeled "SPEC"). This pack will normally be in the drive but it is not required to run SPEC. It is required to run any of the supporting programs discussed in another chapter.

This manual attempts to document the SPEC operating system and related topics. The following section discusses the general operation of SPEC including how to get into the program and how to issue commands. The following chapters discuss the commands in detail, operation of the microprocessor based pulse programmer, spectrometer peripherals, supporting programs, and how to create and load a "TEMP" program. The appendices contain details on bootstrapping procedures, hardware specifics, a technical description of SPEC and how to change or recreate it, a description of microcode loading and the procedure for formatting floppys, disks and recreating RDOS.

II. THE SPEC OPERATING SYSTEM

In this chapter the basics necessary to operating SPEC effectively are presented. Instructions on how to get from RDOS to SPEC, the operating modes of SPEC and how to enter commands into SPEC are given.

A. GETTING FROM RDOS to SPEC

It is first assumed that the computer is currently running in the DG operating system RDOS. If not, then RDOS must be "bootstrapped" by one of the procedures discussed in the appendix. If the disk drive is off, it must be readied. First, turn on the drive power if it is off. At this point, the "LOAD" light should be on. Turn the switch labeled "LOAD/RUN" or "LOAD/READY" to "RUN" or "READY" and wait until the "READY" light comes on. With the terminal on, type a CR (carriage return). If RDOS has previously been booted, then you should get an "R" as a prompt. If a ">" appears, the computer is already executing SPEC and waiting for command input. If "PARTITION IN USE- TYPE C TO CONTINUE" appears, then the computer has already been bootstrapped and you can proceed to get into RDOS (see appendix). Once in RDOS, there are several ways to get into SPEC. If the "SPEC" removable pack is in the drive, simply type DIR SPEC;SPEC or INIT SPEC;SPEC:SPEC. If a different pack is in (you'll know if you get an error when you try above commands) type DIR DPOF;SPEC. (All of these strings are terminated with a CR). All three methods execute the initializing program which sets up the memory partitions for SPEC and presets certain global constants (see appendix). If the display scope is not already on, turn it on. Something should already appear on the screen: this is whatever was in the data buffer when SPEC was last exited. The SPEC prompt ">" should appear on the TTY. SPEC is now in input mode and ready to accept a command. Before going on to describe the three operation modes of SPEC, the general rules for string input are discussed below.

B. STRING INPUT

SPEC has its own teletype drivers and interpreters which recognize certain input as special. All alphabetical characters are uppercase; lowercase letters are converted to uppercase on input and echoed as such. The TTY interface has a one character buffer. If, during the execution of any command except one of the acquisition commands, a character is typed on the console, that character will be echoed when the command is done and SPEC returns to input mode.

With only a one character buffer, only the last character typed is retained. During execution of one of the acquisition commands, TTY input is interpreted as an interrupt and the character is neither retained nor echoed.

Whenever a string of characters is input from the console, whether it is a command, title input, etc., the string is terminated with either an ESCAPE or CR. The only difference between these is that an ESCAPE is echoed as a $ character and a CR echoes as a carriage return/line feed combination. Before a terminator is typed, the previous character may be deleted with the RUB character which erases the previous character and returns the TTY cursor to its position. An entire line of input may be deleted with a CNTL/C which backspaces off all the characters. After a CNTL/C, SPEC is left awaiting new input. Two control characters, CNTL/A and CNTL/D are available for aborting various operations. Their meaning varies depending on when they are typed.

SPEC is capable of interpreting numeric input as either decimal, octal, or hexadecimal. Unless specified otherwise, all numeric input is assumed to be decimal with special code characters appended to the input to indicate another base. If a "D" is the last character on the number, then this explicitly indicates that the input is to be interpreted as a decimal integer. Similarly, if an "O" is used, the number is interpreted as an octal integer. The character "H" is used for hexadecimal interpretation. This same convention will appear throughout this manual.

C. OPERATION MODES

SPEC has three distinct modes of operation each of which will be discussed next. Input mode refers to the situation where the computer is idle and awaiting a new command. Execution mode refers to the actual execution of a command. Acquisition mode is a special case of execution mode in which the computer is acquiring data from the acquisition system and interacting with the pulse programmer. Different methods exist for going from one mode to another.

INPUT MODE

Whenever the SPEC prompt ">" appears as the last character on the TTY, the computer is awaiting a new command from the operator. The general rules applying to the input of any command are given here. The command format is

COMMAND NAME _ PARAMETERS $ or CR

where _ represents a necessary space. The command name is one of the 60 commands listed in the next chapter. The first six characters of this name

are significant. If SPEC is unable to match the command name against the list of commands in its command table, an error is output and SPEC returns to input mode.

In the command line above PARAMETERS represents a string of up to 10 numeric constants. The number of parameters required and their significance depends on the particular command. The parameters are delimited by either a space or a comma. Extra spaces are insignificant. Any of the parameters may be defaulted by simply omitting them from the command line. For example

XAVE 1,20,0,1,1024,512,1024$

defaults none of the parameters for the XAVE command while

XAVE ,20,,1,1024$

defaults parameters #1,3,6 and 7. The string

XAVE $

defaults all the XAVE parameters. When defaulted, parameters take on the values listed with the command description in the next chapter.

When the command line is terminated (by $ or CR) the command processor interprets the command and checks the validity of the parameters. If there are no errors in the command line, SPEC begins execution of the command (enters execution mode). If any of the parameters are in error a message is output and SPEC returns to input mode. At any time during the command input, SPEC can be returned to input mode (producing a ">") by typing CNTL/A. This effectively kills the input line. Except for macro execution and acquisition commands, once a command has been entered and its execution started, there is no way to stop it before normal completion short of rebooting the computer. If a CNTL/D is typed during input mode, the computer is booted and "FILENAME?" appears on the TTY. This effectively kills SPEC and provides the means for returning to RDOS.

EXECUTION MODE

There are two types of command execution in SPEC: console input execution and macro execution. Console input execution is the execution of a command initiated by the input mode described above. Except for a few commands, during execution the display will be blank. Completion of execution is indicated by a return to input mode with the prompt ">" appearing on the TTY. Macro execution refers to command execution initiated with a ME command from the console. The execution of any macro may be stopped before its normal completion by typing CNTL/A. A CNTL/D input will halt the currently executing macro and boot the computer, returning to "FILENAME?". Macro commands are described in more detail later.

ACQUISITION MODE

Acquisition mode is a special case of execution mode. This mode is initiated by one of the acquisition commands RA,CA,SS or TRRA either from the console during input mode or from a macro. During the execution of one of these commands striking any key on the console temporarily interrupts the data acquisition in progress. SPEC returns to input mode and outputs a prompt. Any command except another acquisition command or NFID may then be entered. Upon completion of this command SPEC returns to acquisition mode and continues data collection where it left off. If the ST command is used then SPEC halts acquisition and continues with the next operation which is a return to input mode if the acquisition command came from the console or the next command if it came from a macro string. Acquisition mode may also be halted with a CNTL/A or CNTL/D. From a macro a CNTL/A will halt the macro and return to input mode. A CNTL/D will kill SPEC, boot the computer and output "FILENAME?". If at any time during acquisition mode the computer seems to have "hung up" and the pulse programmer is halted (error light on), try typing a CNTL/F. This is the control character that the computer expects from the microprocessor. This situation will be discussed in more detail in the chapter on the pulse programmer.

D. MEMORY AND FILE STRUCTURE

When SPEC is initialized (by running "SPEC" from RDOS) the memory is partitioned to allow for program and data space. The bottom 12K of memory is used for program space. The remaining 20K is used for data. Sixteen K of this represents what is called the data buffer(s). The remaining 4K is a temporary area used by the TRRA command and, possibly by TEMP. The 16K data buffer area represents one 8K complex vector to the SPEC routines. Most SPEC commands require you to specify whether they should act on the REAL or IMAGINARY portion of this vector. Throughout this manual, these are referred to as the REAL buffer and IMAGINARY buffer and are collectively referred to as the data buffers or buffer. The context will always make clear the usage. While data is conceptually referred to as complex, some of the mathematical operations of SPEC treat the buffers as separate but equal elements. The command description always makes clear which is the case.

SPEC also maintains several files on the fixed disk. Included in these are four "scratch" records which each hold 8K complex words, or the entire contents of the data buffer. Some commands implicitly use scratch records 1 and 2 and so they must be used carefully to avoid data loss. The fixed disk also contains

800 archive records which each hold 1K complex words with a title and several valuable constants. These are used for long term storage of data. Figure 1 shows these disk files and memory areas. The arrows in that figure indicate available paths for transfer of data and the letters are command names which accomplish the particular transfer shown. A brief description of these commands is given below.

SD (save data) transfers entire contents of the data buffers into one of the scratch records.

CD (call data) transfers entire contents of one of the scratch records into the data buffers.

WR (write) transfers contents of the first 1K (complex) of data buffers into one of the archive records.

RD (read) transfers contents of one of the archive records into the first 1K (complex) of the data buffers.

ERASE (erase) deletes archive records so that they may be reused.

To keep track of where things are going in the archive two pointers are used, one for reading and one for writing. The write pointer indicates the next record past the last archive record written into, but is reset to 1 each time SPEC is run. It may be moved with the WR command. Similarly, the read pointer indicates the next record past the last record read from. Again, the read pointer is reset to record 1 each time SPEC is run and may be moved with the RD command.

In addition to these data files, SPEC also uses several other fixed disk files. All the pulse programs, their titles and default registers are stored on disk. There are 100 pulse program records available. PPLOAD transfers a record from the disk into the microprocessor memory and sets default registers and intializes the microprocessor. PPSAVE reads the current pulse program from the microprocessor memory and writes it into one of the disk files along with a title and default registers. PPERASE erases pulse program disk records and makes them available for reuse.

Macro strings are entered into macro records with the MD command. A total of 100 macros are available for use. Any macro may be listed with the ML command and executed with ME. There are 32 records available for phase box settings which are defined by PHD and loaded with PHSET. Finally, all the overlays are stored in an overlay file on the fixed disk. This file is used by the internal operations of SPEC and is not directly accessible by any of the commands.

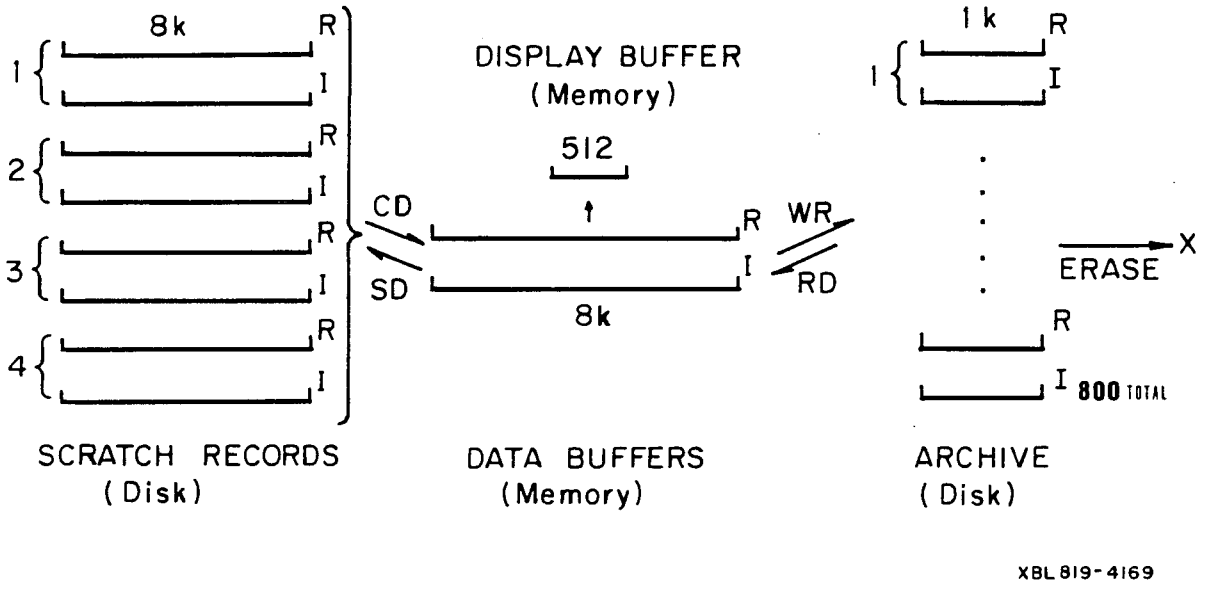

8k
R
I
1
2
3
4
SCRATCH RECORDS
(Disk)
CD
SD
DISPLAY BUFFER
(Memory)
512
R
I
8k
DATA BUFFERS
(Memory)
WR
RD
1k
R
I
1
ERASE
X
R
I
800 TOTAL
ARCHIVE
(Disk)
XBL 819-4169

FIGURE 1

E. OBSERVATION OF DATA

So that data may be observed quickly and easily, data is displayed on an oscilloscope (either Tektronix or HP). The data plotted is stored in a special display buffer. The specified points displayed are chosen by the BUF and LIM commands. BUF determines whether the real or imaginary buffer is displayed and LIM sets the highest and lowest points to be displayed. Initially, when SPEC is started, the first 1K of data points in the real buffer are displayed. Independent of the points selected by LIM, 512 points are actually displayed on the screen. If the number of points selected by LIM is less than 512, linear interpolations are done between actual points so that a total of 512 points are still displayed. If the number of points selected is greater than 512, a subset of 512 of these will be taken (almost) evenly spaced, and this subset is displayed. The points which are intensified in the display are cursor points and are always displayed. If a cursor is moved, the display will be updated again to include the cursor point. The CU command allows the cursor points to be moved, their coordinates to be listed, and may be used to search for peaks. To adjust the display so that the highest point is the maximum possible and the lowest is the minimum possible, the NORM command may be used. The data buffer may be multiplied by a fraction with the SC command. If just the display is to be multiplied by a fraction, but the data buffer is to be left unchanged, the SF command may be used. The buffer and limits chosen for display also determine the current default parameters for many of the SPEC commands.

The data observed on the screen may be plotted on an incremental plotter with the COMPLOT command. Two options exist for plotting. For the first, in which only one page is used, the data points plotted are chosen in a manner similar to the display routine. For the Houston instruments plotter, a total of 1000 points are actually plotted. With the second option, all data points, regardless of number, are always plotted. In this case, plotting is done with the x axis extending down as many pages as necessary to plot every point.

F. ACQUISITION OF DATA

Data may be acquired with several commands, in conjunction with the pulse programmer. Use of the pulse programmer itself will be discussed elsewhere. Assuming that the proper pulse program has been loaded and that the pulse programmer has been suitably activated, the SS command will acquire data without

maintaining any sort of average. ST stops acquisition of data started by any of the acquisition commands. RA will start a stable (Hewlett-Packard) average of incoming data. The number of shots taken is tallied and this number is printed out if averaging is stopped by the ST command. The resulting average is stored in scratch record 1. During acquisition, the average (16 bits) is stored in the first part of the data buffers. The acquisition system converts each shot to 10 bits of digital data and stores it in the data buffers starting at point 4097. If averaging started by RA is stopped by ST, then it may be continued from that point with the CA command. In addition, this command may be used after reading data in from an archive record with the proper mask to reset acquisition parameters (see RD). For most of the standard pulse programs, the dwell time for sampling is set by TI. The minimum is 3 μsec, setable in 1 μsec intervals with a maximum of 1638 μsec. The frequency of display updating during acquisition is set by DMODE.

All the acquisition commands, except TRRA, allow the collection of an FID up to 4K points long, in increments of 256 points when the standard pulse programs are used. The TRRA command allows the collection of an arbitrary length FID (up to 8K) or blocks of data of arbitrary size.

G. DATA PROCESSING

Since much of the data taken is free induction decays, SPEC provides commands for the preparation of and execution of Fourier transforms. Commands for preparation include BC (baseline correction-removal of DC offset), AP (apodization-multiplication by a decreasing triangular weighting function), CL (clear-set to zero region of points), and SHFT (shift-moves data right or left in data buffer). At this point phase corrections may be applied by the MX command or a magnitude spectrum may be generated with the MAG command. Inverse Fourier transformation of data is possible with the IFFT command.

H. PULSE PROGRAMMER COMMANDS

SPEC has a collection of commands that allow one to operate the microprocessor based pulse programmer found on each spectrometer. PPLOAD and PPSAVE allow one to load pulse programs from the disk and save pulse programs in the programmer memory, respectively. Pulse programs may be defined and edited with PPTALK command which handles communication between the console and microprocessor.

Pulse program registers may be set with the PPSET command. In addition, certain special registers are set by the PPLS and PPLI commands. These are the PP loop set and PP loop increment commands.(The reference to "loops" is a result of earlier pulse program hardware.) Some registers are implicitly altered by the NFID, RDLY, TI, NINETY, and acquisition commands.

I. BUFFER ARITHMETIC COMMANDS

There are four commands which allow arithmetic operations to be performed between the data buffer, and any of the scratch records. In each case the result is left in the data buffer with the contents of the scratch record left unchanged. The commands are BADD, BSUB, BMUL, BDVD which perform, respectively, addition, subtraction, multiplication and division. For each of these commands the data buffer points are treated not as complex numbers with a real and imaginary part, but rather as pairs of points. Either member of a pair, or both members may be used in the arithmetic. Thus, BMUL does not form the product (a+ib)*(c+id) but rather a*c or b*d or both a*c and b*d for every point specified. Complex multiplication would require a combination of BMUL, BADD, BSUB and MX commands.

J. OTHER ASSORTED COMMANDS

The following list describes briefly some of the other commands in SPEC. A more complete list and description follow in the next chapter.

ADD This command is used to add a constant value to any portion of the data buffers.

AVE This command collects the average y value of any set of contiguous data points in the data buffer and, if desired, will subtract from this the average of any other set of contiguous data points.

INTR This command performs an integration of y values of points in the data buffer. It is displayed such that the y value of a point in the result is the integral from the left most limit of integration up to that point.

LEV This command sets any region of the data buffers to a specified level.

SM This command performs a three point smooth on any region of the data buffer. This, after many smooths, is equivalent to a Gaussian convolution.

TEMP This command starts execution of an arbitrary program loaded into the "TEMP" overlay. This allows one the ability to write specialized routines which may be made an integral part of SPEC.

XAVE This command calculates an average as with the AVE command but on any archive record or contiguous block of archive records. This series

of averages is loaded into the data buffers using one point for each archive record averaged.

K. CONVENTIONS FOR BUF, LOW, HIGH PARAMETERS

For many commands the region of the data buffers to be operated upon must be specified. This is done with BUF, LOW, HIGH parameters. The following description applies to all commands containing BUF, LOW, HIGH parameters.

BUF indicates which buffer(s) is (are) to be operated upon and may take on the values 0,1,2, meaning:

BUF = 0 both real and imaginary
= 1 real buffer only
= 2 imaginary buffer only.

LOW and HIGH indicate the lowest and highest points to be operated upon in the buffer(s) indicated by BUF.

LOW or HIGH

< 0 specifies current low or high cursor point.
=0 specifies LOW = 1 or HIGH = 8192 respectively.
> 0 indicates LOW or HIGH point number from 1 to 8192 in buffer BUF.

For many commands the defaults for these parameters are specified to be the "current BUF, LOW,HIGH." These are the currently displayed limits in the currently displayed buffer only. Exceptions are found in the command descriptions in the following chapter.

L. MACRO INSTRUCTIONS

As mentioned before, SPEC allows one to define a series of commands in a macro string. Macro strings are stored on disk and may be executed anytime with the ME command. Each macro may be executed once or repetitively. Macros may also be "nested" by including an ME command within a macro string. In theory macros may be nested to any depth. In practice, however, most operations require nesting only a few levels deep. Technically, the only limitation on nesting is the variable stack size. This is currently 4K words and so should allow ample nesting capabilities. Macros are define with the MD command and are listed with the ML command.

In addition to these characteristics, SPEC's macro structure also allows for variable substitution at execution time. In any macro definition, macro variables, specified by an ampersand (&) and a variable number, may replace numeric constants as command parameters. For example, consider a macro defined

as follows (assume this is macro #1):

RD &1$SHFT -1024,0,0,0$MAI 1,&2$

When a macro with such variables is executed, the variables take on the values of parameters in the ME command. Variable number one takes the value of parameter number three and so on. A total of eight variables are allowed taking the values of parameters 3 through 10 in the ME command. If the above macro definition is executed with the command

ME 1,10,100,-1$

then &1 takes the value 100 and &2 takes the value -1.

Two commands are available for setting macro variables to any value and for incrementing them by any value. These are MAS and MAI. MAS sets any of the eight variables to a value and MAI increments them. Both commands must be placed within a macro definition and only affect a variable of the macro they are in. Thus, for the macro defined above, each time it is executed &1 is incremented by &2. If this macro is executed with the ME command above, SPEC will read backward through the archive starting at record #100. The SHFT command places each record read into separate 1K blocks of the data buffer. By changing parameter four in the ME command,SPEC can be made to read records forward or skip some arbitrary number of records between reads. Macro variables are only pertinent to the macro they are in and are not passed to nested macros unless they are included as parameters in the nested ME command.

In addition to the use of macro variables discussed above, they may also be used in an indirect (or nested) addressing manner. This is indicated by using two or more ampersands followed by a decimal number from 1 to 8 in the macro definition. The variable value is determined by continuing to replace ampersands by the value of the parameter pointed to until all ampersands are replaced. As an example, suppose the macro definition contains the following string: (macro #1)

PPLS &&1$

If this macro is executed with

ME 1,1,2,10$

then the definition above is first interpreted as

PPLI &2$

because, with parameter 3 equal to 2, &&1 becomes &2. Finally the definition becomes PPLI 10$.

The sequence of interpretation is then

&&1⇒ &2⇒ 10.

As an example of where this might be useful, consider the situation where one wants to read records from the data archive when the record numbers are not related by a simple increment (for example, records 10,12,25,73,82). This could be accomplished with either of the two macros below:

Macro #2

RD &1$←RD &2$←RD &3$←RD &4$←RD &5$

executed by

ME 2,1,10,12,25,73,82$

or, Macro #3

RD &&1$MAI 1,1$

executed by

ME 3,5,2,10,12,25,73,82$

On the first pass of macro #3, variable 1 takes on the value of 2 and record 10 is read. Variable one is then incremented so that on the next pass it points to parameter 5 and record 12 is read.

This type of nesting can be extended to more than two ampersands, but the results of interpretation are more complicated. For example

RD &&&1$ MAI 1,1$

executed with

ME 3,5,2,3,4,5,10,12$

would result in records #4,5,10 being read, followed by parameter errors for the final two executions.

M. ERROR MESSAGES

If an incorrect command name is entered, an error that the command was not found will be output by the command processor. If parameters are entered in an incorrect format or have values out of bounds, an error will be output by the command's parameter routine. Any other error found (eg. overflow) will cause a message to be output by the routine which detected the error. For data manipulation commands which detect integer overflow, the overflow error is not fatal and overflowed points are set to the extreme integer ranges.

N. LEAVING SPEC

When all operations in SPEC are finished, you may leave SPEC and re-boot the computer by typing CNTL/D while in input mode. The last contents of the data buffers are stored on disk and will be restored to the data buffers next time SPEC is run. A CNTL/D is echoed as "^D" and the computer will eventually respond with "FILENAME?"

In this chapter we have presented the basic operations of SPEC. The following chapter is an index of all the SPEC commands with both a brief and detailed description. Further chapters document the operations of the pulse programmer, a discussion of supporting programs for SPEC run under RDOS, and details on using the TEMP command. The appendix contains information on bootstrapping, initialization, peripherals, a technical description of SPEC and instructions for recreating, backing up, or changing SPEC.

III. SPEC COMMANDS

There are a total of 60 commands in SPEC which control the spectrometer and manipulate data. Commands are either entered from the keyboard or executed from a macro definition as described in Chapter II. This chapter contains a description of each of these commands. A short description of each command is first presented, followed by a detailed description. This detailed documentation includes the command name, title, input format, and functional and operational description. The significance, ranges and defaults of the parameters relevant to each command are also given. Certain default values, such as "Current BUF, LOW, HIGH" are described in Chapter II. Some command descriptions contain warnings on the possible outcome of their execution. These in no way document all the things that can go wrong, but attempt to point out where some difficulties may arise.

Numerical constants given in the descriptions may be appended with a base code letter in the manner discussed in Chapter II. A "D" indicates decimal base, an "O" octal, and an "H" hexadecimal. All numerical parameter ranges are presented for a 32K Word version of SPEC.

SHORT DESCRIPTION OF COMMANDS

1. ADD -adds a specified value to data buffer points.
2. AMODE -sets the current acquisition and averaging modes.
3. AP -performs a triangular apodization of data points.
4. AVE -calculates an average of data points.
5. BADD -adds data points and scratch record.
6. BC -baseline corrects data points.
7. BDVD -divides data points by a scratch record.
8. BMUL -multiplies data points by a scratch record.
9. BSUB -subtracts a scratch record from data points.
10. BUF -sets current buffer.
11. CA -continues averaging process with data in scratch record 1.
12. CD -reads a scratch record into the data buffers.
13. CL -clears data points.
14. COMPLOT -plots data on the Houston Instruments DP1 incremental plotter.
15. CU -enters cursor control mode.
16. DMODE -sets display update mode during acquisition.
17. ERASE -erases data from archive records.

18. FFT -performs a fast fourier transform of data points.
19. FFTO -reports and sets value of FFT parameter.
20. IFFT -performs a fast inverse fourier transform of data points.
21. INTR -integrates data points.
22. LEV -sets data points to specified level.
23. LIM -sets current display limits.
24. MAG -calculates the magnitude of data points.
25. MAI -increments value of a macro argument.
26. MAS -sets the value of a macro argument.
27. MD -defines a macro.
28. ME -executes a macro.
29. ML -lists a macro.
30. MX -performs constant and linear phase shifts of data points.
31. NFID -reports and sets value of NFID parameter.
32. NINETY -sets the value of a 90 degree pulse.
33. NORM -normalizes data points.
34. PHD -defines a phase controller record.
35. PHSET -sets phase controller.
36. PPERASE -erases a pulse program record.
37. PPLI -increments pulse program loop registers.
38a. PPLOAD -loads pulse program, for use with Ackerman microcode.
38b. PPLOAD -loads pulse program, for use with Drobny microcode.
39. PPLS -sets pulse program loop registers.
40a. PPSAVE -writes a pulse program to disk, for use with Ackerman microcode.
40b. PPSAVE -writes a pulse program to disk, for use with Drobny microcode.
41. PPSET -sets a pulse program register.
42. PPTALK -handles operator I/O with pulse programmer.
43. PULSE -displays a pulse wave of specified period, duty, and phase.
44. RA -runs averaging acquisition.
45. RD -reads an archive record into data buffers.
46. RDLY -sets pulse program recycle delay register.
47. SC -scales data points by a constant.
48. SD -stores data buffers in a scratch record.
49. SF -sets display scale factor.
50. SHFT -shifts position of data points.
51. SINE -generates and displays a sine wave.

52. SM -smooths data points.
53. SS -starts single shot acquisition.
54. ST -stops acquisition process.
55. SWORD -reports and sets value of SWORD parameter.
56. TEMP -executes user defined TEMP program.
57. TI -sets acquisition dwell time.
58. TRRA -starts transverse averaging acquisition.
59. WR -writes data buffers to an archive record.
60. XAVE -cross averages archive records.

THE COMMANDS IN DETAIL

1. ADD [ADD CONSTANT VALUE]

Format: ADD VAL, BUF, LOW, HIGH

For the buffer specified, or both, VAL is added to each data point. If overflow occurs then the data point(s) will be set to ±32767, and an overflow message will be output. VAL must be between ±32767.

Defaults: VAL = 0, current BUF, LOW, HIGH

Ranges: VAL = ± 32767

2. AMODE [SET ACQUISITION MODE]

Format: AMODE MODE1, MODE2, MODE3, MODE4

This command determines the disposition of data acquired with each shot. Pulse program control and averaging are determined by the values of the modes. As each shot is taken by an acquisition command (eg. RA), the modes are executed cyclically depending on the shot number. Control and averaging are determined by the codes for MODE# below:

MODE# = 1 PP started at line 01 and new shot is added to average.

MODE# = 2 PP started at line 02 and new shot is subtracted from average.

MODE# = 3 PP started at line 03 and a positive $\pi/2$ shift is performed to new shot before it is added to average.

MODE# = 4 PP started at line 04 and a positive $\pi/2$ shift is performed to new shot before it is subtracted from average.

This control of acquisition allows a single pulse program to be used for such things as phase cycling of the r.f. pulses.

Defaults: MODE1 = 1 MODE2 = MODE3 = MODE4 = NULL

Ranges: MODE# = 1-4, NULL

Warnings: Pulse program should contain BR statements for each of the modes used to transfer control to the appropriate section. The number of non-null modes determines the period of the cyclic acquisition.

3. AP [APODIZE]

Format: AP BUF, LOW, HIGH

For each buffer specified, the average value between LOW and HIGH is subtracted from each data point. If overflow occurs, values are set to $\pm$ 32767 and an overflow message is output to the console. The points from 1 to LOW -1 and from HIGH +1 to NFID (if HIGH < NFID) are set to zero. Triangular apodization is then performed starting at point LOW and ending at point HIGH for each buffer.

Defaults: Current BUF, LOW,HIGH

4. AVE [CALCULATE AVERAGE VALUE]

Format: AVE BUF, LOW1, HIGH1, LOW2, HIGH2

For each buffer, the average value of the data points from LOW1 to HIGH1 is calculated and the average value of points from LOW2 and HIGH2 is subtracted from this. LOW2 and HIGH2 equal to null omits the subtraction step. The average is then output to the console. If both buffers are specified (BUF = 0) the averages are output as real buffer average, then imaginary buffer average.

Defaults: Current BUF, LOW, HIGH, NULL, NULL

5. BADD [ADD BUFFER AND SCRATCH RECORD]

Format: BADD SREC, BUF, LOW, HIGH

This command adds, point by point, scratch record SREC to the date buffers specified. Points added from SREC and buffers are specified by LOW, HIGH. Result is returned to data buffer and checked for overflow. Overflowed points are set to $\pm$ 32767 and a message is output to the console.

Defaults: SREC = 1, current BUF, LOW HIGH

Ranges: SREC = 1-4

6. BC [BASELINE CORRECT]

Format: BC BUF, LOW1, HIGH1, LOW2, HIGH2

This command removes a baseline offset from the specified data buffers. The offset is calculated as the average of data points from LOW2 to HIGH2 in the data buffer. If LOW2 and HIGH2 are omitted, then the average is calculated from LOW1 to HIGH1. This offset is then subtracted from data points between LOW1 and HIGH1. The result is checked for overflow. If detected, overflow points are set to $\pm$ 32767 and a message is output to the console.

Defaults: Current BUF, LOW, HIGH, NULL, NULL

7. BDVD [BUFFER DIVISION BY SCRATCH RECORD]

Format: BDVD SREC, BUF, LOW, HIGH

This command divides,point by point,the data buffer(s) specified by the scratch record SREC. The points involved in both data buffer and SREC are LOW to HIGH. The result is the quotient multiplied by 32767 and is checked for overflow. If detected, the overflowed points are set to $\pm$ 32767 and a message is output to the console. The result is left in the data buffer.

Defaults: SREC = 1, Current BUF, LOW, HIGH

Ranges: SREC = 1-4

8. BMUL [BUFFER MULTIPLICATION BY SCRATCH RECORD]

Format: BMUL SREC, BUF, LOW, HIGH

This command multiplies,point by point,scratch record SREC and the specified data buffer(s). The points from LOW to HIGH in the data buffer receive the result. The result is the high order word of the double precision integer multiplication (ie. the result is interpreted as the product divided by 32767).

Defaults: SREC = 1, Current BUF, LOW, HIGH

Ranges: SREC = 1-4

9. BSUB [SUBTRACTION OF SCRATCH RECORD FROM BUFFER]

Format: BSUB SREC, BUF, LOW, HIGH

This command subtracts, point by point, scratch record SREC from the specified data buffer(s). The points involved are those from LOW to HIGH. The result is left in the data buffer and is checked for overflow. If detected, overflowed points are set to $\pm$32767 and a message is output to the terminal.

Defaults: SREC = 1, Current BUF, LOW, HIGH

Ranges: SREC = 1-4

10. BUF [SET CURRENT DISPLAYED BUFFER]

Format: BUF IBUF

This command sets the buffer to be currently displayed. Also set is the "current BUF" default of all commands excepting this default. IBUF = 1 specifies the real buffer; IBUF = 2 specifies the imaginary buffer.

Default: Buffer not currently displayed (ie. switch)

11. CA [CONTINUE AVERAGING]

Format: CA NSHOTS

This command continues averaging of acquired data that was started by the RA command and subsequently stopped by ST. The current average is taken from scratch record 1 and the current number of shots is noted. NSHOTS more are taken, then averaging stops automatically. If NSHOTS = 0 the averaging continues up to 32766 total shots. NSHOT must be less than 32766.

Default: Averaging continues up to previous value of NSHOTS set by RA.

Ranges: NSHOTS = 0-32766

12. CD [READ DATA FROM SCRATCH RECORD]

Format: CD SREC

Data is read from scratch record SREC into the data buffers. Contents of the scratch record are not affected. Limits of display are not changed.

Default: SREC = 1

Range: SREC = 1-4

Warning: The entire scratch record is read and previous contents of data buffer are lost.

13. CL [CLEAR BUFFER]

Format: CL BUF, LOW, HIGH

This command sets to zero all points in specified data buffer(s) from LOW to HIGH.

Defaults: Current BUF, LOW, HIGH

14. COMPLOT [PLOT ON COMPLOT PLOTTER]

Format: COMPLOT NCOPY, IFRM, MODE

This command plots the currently displayed data ("display buffer") on the Houston Instruments DP1 COMPLOT plotter. NCOPY determines the number of copies that will be made. IFRM determines the pen and paper position after plotting is finished and is described below. There are two types of plots available which are chosen by the value of MODE.

MODE = 1

For this mode the plot is on a single 8½ by 11 inch piece of paper with the long dimension the x-axis. The pen should be started about ¼ inch from the top edge and about ½ inch from the left edge of the paper. The actual plot dimensions are 8 by 10 inches. A total of 1000 points are plotted in the x direction and the y-axis is an 800 point vertical scale. Data points plotted are those between the current LOW and HIGH values. If this number of points is less than 1000, linear interpolation is done as needed. If the number is greater than 1000, then data points are dropped in an (almost) evenly spaced fashion.

For IFRM = 0 the paper is advanced to the start of the next page. For IFRM = 1 to 800 the pen is returned to the y-axis position equal to IFRM (1 = top, 800 = bottom of page) and the left edge of the plot.

MODE = 2

For this mode the x and y axises are swapped from that of MODE = 1. Thus, the y-axis (1000 points) extends from either edge of the paper and the x-axis runs down the paper over as many pages as necessary. Every point from LOW to HIGH is plotted. The pen should be started ¼ inch from the top and ½ inch from the right side (closest to the on/off switch) of the paper.

IFRM takes the same meaning for this mode except that it may range from 0 to 1000 and if IFRM= 0, the pen is returned to the start of the plot.

For either mode, setting NCOPY= 1 and IFRM = 1 will cause over-plotting and may be used to intensify the plot line.

Defaults: 1,0,1

Ranges: NCOPY = 1-32767, IFRM = 0-800 for MODE = 1; 0-1000 for MODE = 2, MODE = 1-2

Warning: Specifying MODE = 2 for limits of 1 to 4096 will produce a plot of 4.82 pages!

15. CU [CURSOR CONTROL]

Format: CU LCUR, HCUR

This command allows manipulation of the cursors (intensified scope points). Execution of the command sets the cursor positions at the x values specified by LCUR, HCUR. The command then enters a subcommand processor and awaits input from the console. Subcommands are one character keystrokes which allow the user to move the two cursors, find peaks, retrieve x,y coordinates, and set display options. These subcommands are not echoed on the console and may be entered until the K subcommand is issued. This returns control to normal SPEC command operation. Subcommands are:

Control:

H- High cursor
Subcommands following this command which pertain to a cursor operate on the high (upper) cursor point.

L- Low cursor
Subcommands following this command which pertain to a cursor operate on the low cursor point.
Setting H or L means that <u>all</u> the following cursor subcommands will operate on the high or low point, respectively, until another H or L is issued.

K- Kill
Stop cursor command and return to normal SPEC operation.

Cursor:

U- UP

Move specified cursor toward higher points.

D- Down

Move specified cursor toward lower points.

U and D need only be issued once until changed by next U or D subcommand.

0- 10**0

Move specified cursor specified direction 1 point.

1- 10**1

Move specified cursor specified direction 10 points.

2- 10**2

Move specified cursor specified direction 100 points.

3- 10**3

Move specified cursor specified direction 1000 points.

C- Coordinates

Type on console x and y coordinates of last specified cursor.

T- Threshold

Sets threshold for P subcommand to current y value of last specified cursor.

P- Peak

Search in last specified direction for the next local maximum which has a y value $\geq$ to that set by T subcommand. Last specified cursor is moved to this maximum point.

Display:

R- Real

Display real buffer.

I- Imaginary

Display imaginary buffer.

X- Expand

Expand display to the limits set by the current cursor positions.

B- Back

Return the display to the limits set by the previous display limits.

S- Smaller

Sets the display buffer scale factor to ½ of its current value (similar to SF command).

G- Greater

Sets the display buffer scale factor to 2 times its current value.

N- Normal

Sets the display buffer scale factor to 1.

The X and B subcommands may be used to change the limits of display. The X subcommand expands the display and the B subcommand returns it to the previous limits. CU only retains the last previous limits so typing XXB will not change the display limits after the first X whereas XBXB will alternate between previous and current limits.

The subcommands S,G, and N affect the display scale factor. The actual data buffer is not scaled, only the display buffer. On entering CU the display is set at the current scale factor set by SF. Exiting CU, the scale factor is left at whatever value it had after all S,G,N subcommands. The N subcommand is equivalent to SF 1,1$ outside of CU.

Defaults: Current LCUR, HCUR

Note: On entering CU, the lower cursor is set to move up, the "previous" limits for B are the full data buffer, the threshold for T is set to -32767, current BUF,LOW,HIGH are displayed at the current scale factor.

16. DMODE [SET ACQUISITION DISPLAY MODE]

Format: DMODE N

This command allows control of the display update frequency during data acquisition. The display will be updated to show the current average or single shot every Nth shot. If N=0, the display will not be updated at all during acquisition. For the TRRA command, the DMODE parameter pertains to scope update of the temporary buffer. This command does not affect scope update for commands other than the acquisition commands even if they are executed during acquisition.

Default: N = 1

17. ERASE [ERASE ARCHIVE RECORDS]

Format: ERASE LREC, HREC

This command erases the archive records from LREC to HREC, inclusively. The records are then available for use for new data. If HREC is omitted or if LREC = HREC, then only record LREC is erased. Care must be taken in using this command, as it is perfectly capable of erasing the entire archive if told to do so.

Defaults: No defaults

Ranges: LREC = 1-800, HREC = 1-800

18. FFT [FAST FOURIER TRANSFORM]

Format: FFT NFFT

This command performs a Cooley-Tukey fast Fourier Transform of the current data buffers. The first NFFT points of the data buffers are used and must be an integral power of two, less than or equal to 8192. If NFFT is greater than the current FID length (set by NFID or an acquisition command) then zero filling from NFID +1 to NFFT is automatically performed before the Fourier Transformation. After Fourier Transformation, the display limits are set to 1 to NFFT, the scale factor is output to the console and the transformed data is copied into scratch record 2. The scale factor is a power of two and may be interpreted as the factor required to prevent integer overflow during transformation. Small relative signal amplitudes produce more negative scale factor exponents.

Defaults: NFFT = last value set by FFT or FFTO. Initially 1024 on entering SPEC.

19. FFTO [SET FFT PARAMETER]

Format: FFTO NFFT

This command sets and reports the current value of the parameter NFFT. This parameter determines the length of the transform performed by the FFT and IFFT commands. It may also be set by FFT. A NULL for NFFT will cause FFTO to type its current value on the console.

Default: NFFT = NULL

Range: NFFT = Power of two GE 2 and LE 8192.

20. IFFT [INVERSE FOURIER TRANSFORM]

Format: IFFT

This command performs an inverse Fourier Transform on the data buffers with NFFT taking its current value (set by either FFT or FFTO).

Defaults: No parameters to default.

21. INTR [INTEGRATE]

Format: INTR BUF, LOW, HIGH

This command performs a numerical integration on the specified points for the specified buffer(s) and normalizes it so that the first point has its normalized y value and the last point is +32767 or -32767 depending on whether the sum of all points is positive or negative. The integral is displayed as a curve with the y value of a point corresponding to the normalized integral up to that point. Values may be obtained with the CU command. NOTE: for a good integration a flat baseline at zero intensity is needed.

Default: Current BUF, LOW, HIGH

Warning: The points integrated are replaced by the integral curve so that the data is lost.

22. LEV [SET DATA TO VALUE]

Format: LEV VAL, BUF, LOW, HIGH

This command is used to set the y value of all the points in the specified buffer(s) from LOW to HIGH to a value of VAL.

Defaults: VAL = 0, Current BUF, LOW, HIGH

Range: VAL = +32767 to -32767

23. LIM [SET DISPLAY LIMITS]

Format: LIM LOW, HIGH

This command sets the lowest and highest points to be observed to LOW and HIGH. LOW = 0 or HIGH = 0 indicate the extreme limit of the buffer, that is point 1 or 8192 respectively. Using either LOW or HIGH negative replaces its value with the current cursor position, e.g. LOW = -1 means LOW = LCUR. This command can be used during all the acquisition routines to set the limits of observation. During the operation of the TRRA routine, the LIM command operates somewhat differently. If TRRA is stopped from the console during averaging and the display is updated for the TEMP data buffer, then LIM will have no affect and will simply return control to TRRA. See the TRRA command description.

Defaults: LOW = HIGH = -1, current cursor positions.

24. MAG [CALCULATE MAGNITUDE SPECTRUM]

Format: MAG LOW, HIGH

This command calculates a magnitude spectrum between the specified points. The magnitude is calculated as the magnitude of the data buffers as a complex vector. That is, the algorithm is

$$MAG(DATA(I)) = SQRT\ (\ REAL(DATA(I))^{**}2 + IMAG\ (DATA(I))^{**}2)$$

where REAL (DATA(I)) is the real part of the complex data point and IMAG (DATA(I)) is the imaginary part. SQRT is an integer square root function which approximates a real square root by an iterative routine. The result is left in the real data buffer replacing the current data between LOW and HIGH. The imaginary data buffer is unchanged.

Default: Current LOW, HIGH

25. MAI [INCREMENT MACRO ARGUMENT]

Format: MAI IARG, INCR

This command may be used to increment one of the eight macro arguments that are passed to commands within a macro (see ME, MD). The macro argument IARG is incremented by INCR from its current value. The new value is the value which will replace the macro argument variable in the macro. The MAI command must be executed within a macro (ie. it may not be executed as a normal SPEC command from the console and must be entered with the MD command). MAI only increments an argument for the macro within which it resides and will not affect nested macros unless the argument itself is passed with a ME command.

Defaults: IARG = 1, INCR = 1

26. MAS [SET MACRO ARGUMENT]

Format: MAS IARG,IVAL

This command may be used to set one of the eight macro arguments that are passed to commands within a macro (see ME,MD). The value of the argument IARG is set to IVAL. This new value will then replace the macro argument variable in the macro. The MAS command must be executed within a macro (i.e. it may not be executed as a normal SPEC command from the console and must be entered with the MD command). MAS only sets the argument for the macro within which it resides and will not affect arguments in nested macros unless the argument IARG itself is passed in a ME command within the macro.

Defaults: IARG = 1, IVAL = 0

27. MD [MACRO DEFINE]

Format: MD N

This command allows one to enter a string of commands as a macro definition. The definition is written to disk as macro number N and may then be executed with the ME command. Commands are entered exactly as in normal SPEC operation except that parameters may be replaced by macro arguments (below). Commands are terminated with either an escape or a carriage return. All corrections (rubout and CNTL/C) operate only on the command being typed; errors in a command that has been entered with a terminator may only be corrected by redefining the entire macro. A total length of 120 characters (not including terminators) may be input. This is about a line and a half if only escapes are used to terminate commands. Two terminators in a row complete the macro string. A CNTL/A aborts the macro definition and returns to normal SPEC control. Previous contents of a macro are lost with each new definition.

Macro argument variables may be used in place of numeric values as parameters in the macro command string. These are entered as the ampersand (&) symbol followed by a single digit decimal integer from one to eight. For example, the macro string may include the command:

XAVE 1, &1, 0, 0, &2, &3$

The macro argument variables are replaced by the numeric value of the parameters for the ME command for this macro. (These are parameters 3-10,see ME.) For example, in the above macro command, &1, &2 and &3 would be replaced by 100, 20, 30 respectively if it was executed with the command ME N,NTIMES, 100, 20, 30$. Parameters in the ME command may themselves be macro argument variables when macros are nested.

Default: N=1

Range: N= 1-100

28. ME [EXECUTE MACRO COMMAND STRING]

Format: ME N, NTIMES, ARG1, ...,ARG8

This command executes macro #N. The commands in macro #N (defined by MD) are read from the disk and executed in the order defined. The entire macro is executed cyclically a total of NTIMES. The macro argument variables &1, &2,..., &8 within the macro definition are replaced with the values ARG1, ARG2,... ARG8, respectively. These values may be reset and/or incremented within the macro with the MAI and MAS commands. Macros may be nested by placing a ME command in a macro definition.

A macro execution may be interruped from the console. Typing CNTL/A will abort all levels of macro execution and return control to normal SPEC operation. Typing CNTL/D will also abort the macro but will leave SPEC and load the bootstrap from DPO.

Defaults: N=1, NTIMES =1, ARG1=ARG2=...ARG8 = NULL

Ranges: N=1-100, NTIMES =1-32767, ARG1 =...ARG8 = $\pm$32767.

29. ML [MACRO LISTING]

Format: ML N

This command lists on the console the current command string defined as macro N.

Default: N=1

Range: N=1-100

30. MX [MIX = PHASE CORRECTION]

Format: MX PHC, PHL, BUF, LOW, HIGH

This command preforms a phase correction with both constant and linear terms. PHC specifies the value of the constant term and PHL the linear term. These are entered as degrees of rotation in the complex plane. A fresh copy of scratch record 2 is loaded into data buffers and, for the specified buffer(s) and points, the phase correction is performed as follows:

R_i^{old} = y value of i^{th} real buffer point

I_i^{old} = y value of i^{th} imaginary buffer point

Φ_i = PHC + PHL(i-LOW)/ (HIGH - (LOW - 1))

$R_i^{new} = R_i^{old} \cos \Phi_i + I_i^{old} \sin \Phi_i$:

$I_i^{new} = -R_i^{old} \sin \Phi_i + I_i^{old} \cos \Phi_i$:

The result is left in the data buffers and is checked for overflow. If detected, the overflowed points are set to $\pm$ 32767 and a message is output to the console.

Defaults: PHC and PHL last used values, initialized to zero on entering SPEC, current BUF, LOW, HIGH.

Warning: Since MX reads in the contents of scratch record 2, the previous contents of the data buffers are lost.

31. NFID [SET NFID PARAMETER]

Format: NFID NFID

This command may be used to set or report the current value of parameter NFID. This parameter sets the total # of acquisition points for the RA,CA,SS commands and is used by the FFT command to determine the requirements of zero filling. If NFID = NULL, the current value of the parameter is typed on the console. Any other value of NFID must be a multiple of 256. NFID then sets the pulse programmer register B8H to 256/NFID. This register is usually used in pulse programs to determine the number of RAM cycles with 256 sampling masks to be executed. After issuing the "LO" command to the pulse programmer, NFID waits for a CNTL/F prompt. This usually is supplied by the pulse programmer but may also be typed on the console.

Default: NFID = NULL

Note: NFID may not be executed during acquisition. An error is typed out in this case.

32. NINETY [SET NINETY TIME]

Format: NINETY I90

This command sets the pulse programmer register D0H to -I90. Thus ITIME is in the units of 0.1 μsec for a delay using D0. This register is usually used in pulse programs to define the duration of a ninety degree pulse (hence the name). After issuing the "LO" command, NINETY waits for a CNTL/F prompt which usually is supplied by the pulse programmer but may also be typed on the console. I90 is also stored and its current value is used when a new pulse program is loaded from disk and initialized.

Default: I90 = 50 (5.0 μsec)

Range: 1 - 16384 (The longest delay which can be set with a single PP register is 1.6384 msec.)

33. NORM [NORMALIZE]

Format: NORM BUF, LOW, HIGH

This command does a vertical expansion of specified points such that the maximum point in specified region is + 32767 and the minimum is -32767. All proportionalities are conserved.

Defaults: Current BUF, LOW, HIGH

34. PHD [DEFINE PHASE SETTINGS FILE]

Format: PHD REC, NPHASE

This command is used to define a file containing phase settings for the phase box controller. REC is the number of the record and NPHASE is the number of settings that are to be entered. After entering the command, the program will prompt for input. The phase settings can then be entered as octal numbers, terminated by a carriage return or an escape. Each setting is checked and if it is out of bounds an error will be output to the console. The value may then be reentered. After NPHASE settings are entered, the file will be written to disk and the record number typed out to the console.

Defaults: REC = 1, NPHASE = 1

Ranges: REC = 1-32, NPHASE = 1-1024

Warning: No check is made on the availability of a phase record. In writing a new record to disk, PHD destroys the contents of the old record.

35. PHSET [SET PHASE CONTROLLER]

Format: PHSET REC

This command loads the RAM memory of the phase controller via its interface with the computer. Record number REC of previously defined settings (PHD) are read from disk. A total of 1024 eight bit settings are output to the RAM. These are loaded in the same order they were entered. The RAM address counter is reset on exit.

Default: REC = 1

Range: REC = 1-32

36. PPERASE [ERASE PULSE PROGRAM RECORD]

Format: PPERASE LREC, HREC

This command erases pulse programs from the pulse program archive, making their disk space available for new pulse programs. If HREC = LREC then just the single record specified is erased. Specifying LREC = 0 means LREC = 1. Likewise HREC = 0 means HREC = 100. Omitting HREC means HREC = LREC. Caution must be used with this command as it is capable of erasing the entire PP archive if told to do so.

Defaults: No defaults

37. PPLI [PULSE PROGRAM LOOP INCREMENT]

Format: PPLI VAL1,VAL2,VAL3,VAL4

This command is used to increment the current values of the pulse programmer "loop" settings. The current loop values are incremented by the values of VAL1,...,VAL4. The updated values are then loaded into the pulse programmer registers B0, B2, B4, B6 (hexadecimal), respectively. Finally, the new values are output to the console. These four registers are often used to define loop counters within the pulse program, hence the command name. Their exact use depends on the particular pulse program currently loaded, however, and reference should be made to the program's write-up to interpret the meaning of these registers. Supplying zero for any of the parameters means that its current value remains unchanged. Zero for all parameters just produces a type out of the current loop values. Current loop values are loaded with each new pulse program.

Defaults: VAL1=VAL2=VAL3=VAL4=0

Ranges: VAL1 to VAL4 = 0 to ±32767

Warning: An attempt to increment a loop value below 1 or above 32767 will cause an error to be output.

38a. PPLOAD [LOAD PULSE PROGRAM], for use with Ackerman microcode

Format: PPLOAD REC,LNH

This command reads from disk pulse program record #REC and loads the pulse programmer microprocessor. After loading, the pulse program is commanded to start execution beginning at line number LN of the program. This can be used to initialize the programmer (eg. some standard programs can be initialized for sampling from the RAM by letting LN take its default value). If LN is less than or equal to zero, the initialization procedure is omitted. In addition to register values stored on disk with the pulse program (see PPSAVE) the following registers are set to default/current values and loaded into the microprocessor.

Register (Hexadecimal)	Value (Decimal)
B0	PP LOOP 1
B2	PP LOOP 2
B4	PP LOOP 3
B6	PP LOOP 4

Register (Hexadecimal)	Value (Decimal)
B8	NFID/256
BA	IRDLY
D0	I90
D2	10*(-ITIME)
A0	0
A2	1
A4	10
A6	100
A8	255
AA	1000
AC	127
C0	-1000
C2	-2500
C4	-10000

Defaults: REC = NULL, LN = A0
Ranges: REC = 1-100, LN = 01-FF_{16}

38b. PPLOAD [LOAD PULSE PROGRAM], for use with Drobny microcode

Format: PPLOAD REC1,REC2,REC3,REC4,LNH

This command reads from disk up to 4 different pulse program records (numbers REC1,REC2,REC3,REC4) and loads the pulse programmer microprocessor. After loading, the pulse programmer is commanded to start execution at line number LN of the program. This can be used to initialize the programmer. If LN is less than or equal to zero, the initialization procedure is omitted. The microprocessor is loaded with those register values stored on disk with REC1 only (see PPSAVE), thus it is important to load two or more pulse programs in the same order in which they were saved. Registers B0-BA, D0, and D2 are set to current parameter values as described at the end of command 38(a).

Defaults: REC1=REC2=REC3=REC4=NULL, LN = A0

Ranges: REC1 to REC4 = 1-100, LN = 01-FF_{16}

39. PPLS [PULSE PROGRAMMER LOOP SET]

Format: PPLS VAL1, VAL2, VAL3, VAL4

This command is used to set the current values of the pulse programmer "loop" settings. VAL1,...,VAL4 are stored in memory as the new current values. The pulse programmer registers B0, B2, B4, B6 (hexadecimal) are loaded with VAL1, VAL2, VAL3, VAL4, respectively. These four registers are often used to define loop counters within the pulse program, hence the command name. Their exact use, however, depends on the current pulse program and reference should be made to the program write-up for details. Omitting any of the values retains their previous values.

Defaults: VAL1 = VAL2 = VAL3 = VAL4 = Current values

40a. PPSAVE [SAVE PULSE PROGRAM ON DISK], for use with Ackerman Microcode

Format: PPSAVE REC

This command obtains from the pulse programmer microprocessor the current pulse program and its default registers and writes them to the pulse program archive in record #REC. The command will not overwrite an existing pulse program in a record. If REC is full then the pulse program is written to the next available record. If the entire archive is full, an error is output and PPSAVE returns to SPEC. Once an available record has been found, PPSAVE prompts for a title which may be up to 78 characters and is terminated by either a carriage return or an escape. After the record is written, the record number is typed out on the console. The current values of the microprocessor registers 80H to FEH are stored with the program on disk.

Defaults: REC = 1

Range: REC = 1-100

Warning: Do not strike any keys on the console after typing in the title and before receiving the SPEC ">" prompt. During this time, the microprocessor is dumping its memory to the NOVA via the console.

40b. PPSAVE [SAVE PULSE PROGRAM ON DISK], for use with Drobny Microcode

Format: PPSAVE REC1, REC2, REC3, REC4

This command is similar to that described in 40(a)--that command description should be read first. In fact, if the pulse program to be saved has 64 or fewer lines, only REC1 need be specified and the description in 40(a) is sufficient.

There are a couple of basic differences if more than one record is specified. In this case, REC1 stores line numbers 01-3F, regardless of the number of actual program lines contained within these limits. Similarly, REC2 stores lines 40-7F, REC3 80-BF, and REC4 C0-FF. PPSAVE prompts for a separate title for each specified record. However, the contents of registers 80H thru FEH are stored on disk with REC1 only. This becomes an important consideration when two or more records are read back in using the PPLOAD command (which see).

41. PPSET [SET PULSE PROGRAMMER REGISTER]

Format: PPSET REGH, VAL

This command loads the pulse programmer microprocessor register REG with VAL. Previous contents of the register are lost. After loading the register, PPSET waits for a CNTL/F prompt from the microprocessor.

Defaults: REG = NULL, VAL = 0

42. PPTALK [COMMUNICATES WITH PULSE PROGRAMMER]

Format: PPTALK

This command allows the operator to communicate with the Pulse Programmer microprocessor (UP). The communication is through the TTY and is handled by the NOVA. Entering PPTALK produces a prompt (% for Ackerman microcode, CMD> for Drobny microcode (see Sec. VII)) which indicates that the program is ready to receive instructions. Instructions are typed in following the normal SPEC string input. You may use rubouts or CNTL/C. Either CNTL/A or CNTL/D exits PPTALK and returns to normal SPEC operation. A complete list of commands which may be executed from PPTALK is given in Sec. VII on the Pulse Programmer.

A word about one of these commands, LI. Under the Ackerman microcode, when you ask for a listing from the Nova, the computer copies the entire pulse program from the UP into its own memory. Subsequent list commands then access this copy unless you issue a CL or line number command, or leave PPTALK. If you do any of these, a fresh copy of the pulse program will be copied before any further listings can be obtained.

NOTE: When using PPTALK with the Ackerman microcode, you do not have to type the special control characters (CNTL/B and CNTL/E) and the leading space required in stand alone UP operation. PPTALK supplies these to the appropriate command strings automatically.

43. PULSE [GENERATE PULSE WAVEFORM]

Format: PULSE PER, DUTY, PHASE, BUF, LOW, HIGH

This command loads into the specified points and updates the display with a pulse train with period PER, duty cycle DUTY, in percent, phase delay PHASE, in degrees. The high value is +32767 and the minimum is 0.

Defaults: PER = HIGH - LOW + 1, DUTY = 50, PHASE = 0,
current BUF, LOW, HIGH.

44. RA [RUN AVERAGE ACQUISITION]

Format: RA NTIMES, NFID

This command initiates the data acquisition and averaging routines. A total of NTIMES shots are taken (unless averaging is stopped from the console, see below). Averaging of data is done using a Hewlett-Packard stable average. This averaging is calculated point by point with the following algorithm:

$$Y^{AVE}_{NEW} = Y^{AVE}_{OLD} + (Y^{SHOT}_{NEW} - Y^{AVE}_{OLD})/N$$

where N is the power of two nearest, but greater than the actual current number of shots that have been averaged. Before this algorithm is applied the current shot is prepared for averaging according to the specifications set by the AMODE command. For each shot taken, the pulse programmer is started at the appropriate line number in the manner described for the AMODE command. The display is updated after each shot unless otherwise specified by DMODE. Initially the command sets the display limits to 1 to NFID, however, this may be changed during averaging with the LIM or CU commands. NTIMES must be less than or equal to 32766 and NFID must be less than or equal to 4096 and a multiple of 256. NTIMES = 0 causes average to continue to 32766 shots unless otherwise stopped. The current average is stored in the data buffer from point 1 to NFID and the current shot is stored from 4097 to (4097 + NFID -1). Averaging may be interrupted at any time from the console by striking any key (this is not echoed) and the SPEC > prompt will appear. Commands may then be entered and when execution is complete, SPEC will return to averaging. The averaging may be stopped at any time with the ST command. If NTIMES shots have not yet been acquired, then the number of shots actually taken will be output to the console. When averaging is complete

or otherwise stopped, a copy of the data buffer is written to scratch record 1 (the previous contents of that record are lost).

Defaults: NTIMES = 0, NFID = current value.

Ranges: NTIMES = 0 - 32766, NFID = 256 - 4096 (multiple of 256)

NOTES: For high S/N on single shots, no significant improvement in average may be observed after many shots have been taken. After each shot the computer awaits a CNTL/F prompt from the pulse programmer after the recycle delay. For executing commands during averaging that finish before the recycle delay, the computer may need to be prompted from the console in order to continue.

45. RD [RD DATA]

Format: RD REC, MASK

This command reads data from archive record REC into the first 1024 points of the data buffers, with provisions set by MASK below. The limits of observation are set to 1 to NFID and the display is updated. The read pointer (next default record to read) is incremented after each read. The title stored with the data and record number are output to the console unless inhibited by MASK. MASK is entered in decimal. MASK word bits have the following significance:

BIT 15 must be clear to allow data transfer.

BIT 14 must be clear to print title and record number.

BIT 11 must be set to restore pulse program parameters (loop values).

BIT 10 must be set to restore FFT parameters (NFFT and scale factor).

BIT 9 must be set to restore acquisition parameters (NFID, ITIME, current shot number).

BIT 8 must be set to restore mixing parameters (PHC, PHL).

NOTE: If record REC is empty, the contents of the next full record will be read.

Examples using MASK:

RD REC, 1$ just positions read pointer; no data is read.

RD REC, 42$ reads next record without printing title, restoring FFT and mixing parameters.

(MASK = 42D = 101010 (binary))

Defaults: REC = record pointed to by read pointer.
MASK = 0
Ranges: REC = 1 - 800, MASK = 0 - 63 (decimal).

46. RDLY [SET RECYCLE DELAY]

Format: RDLY IRDLY

This command is normally used to set the pulse programmer recycle delay with IRDLY in units of tenth seconds. The command loads the pulse program register BAH with IRDLY. The actual use of this register depends on the current pulse program. After loading, the computer waits for a CNTL/F prompt from the pulse programmer before returning to normal operation.

Defaults: IRDLY = 10

47. SC [SCALE DATA]

Format: SC NUM, DENOM, BUF, LOW, HIGH

This command scales data in specified regions by multiplying each point by NUM/DENOM. The result is checked for overflow and if it is detected, overflowed points are set to $\pm$ 32767. No overflow message is output to the console. Result is left in the data buffer.

Default: NUM = 1, DENOM = 1, current BUF, LOW, HIGH.

48. SD [SAVE DATA]

Format: SD SREC

This command copies the entire contents of the data buffer into scratch record SREC. Previous contents of SREC are lost. NOTE: Some commands implicitly use scratch records 1 and 2; no warning is given.

Default: SREC = 1

Range: SREC = 1 - 4

49. SF [SET SCALE FACTOR]

Format: SF NUM, DENOM

This command scales the display buffer by multiplying each point by NUM/DENOM. If there is overflow, overflowed points are set to $\pm$32767 but no message is output to console. Data buffers are not affected.

Defaults: NUM = 1, DENOM = 1

50. SHFT [SHIFT DATA]

Format: SHFT NPT, BUF, LOW, HIGH

This command moves the set of data buffer points specified to the left or right NPT points. Data is shifted left if NPT >0, right if NPT <0. Zeros are inserted in points vacated by shifting and data is lost when shifted out of specified range. Data outside of LOW, HIGH is not affected. Result is left in data buffer.

Default: NPT = 1, current BUF, LOW, HIGH

51. SINE [GENERATE SINE WAVE]

Format: SINE PER, PHASE, BUF, LOW, HIGH

This command generates a sine wave of period PER, in points, and phase angle PHASE, in degrees, in specified points. Sine wave amplitude is $\pm$32767.

Defaults: PER = HIGH - LOW + 1, PHASE = 0, current BUF, LOW, HIGH

52. SM [SMOOTH]

Format: SM NTIMES, BUF, LOW, HIGH

This command performs a three point smooth on the specified data points NTIMES times. The algorithm used is

$$Y_i^{new} = \frac{1}{4}(Y_{i-1}^{old} + 2Y_i^{old} + Y_{i+1}^{old})$$

between i = LOW and i = HIGH. Points LOW and HIGH are not changed.

Defaults: NTIMES = 1, current BUF, LOW, HIGH

53. SS [SINGLE SHOT]

Format: SS NFID

This command starts acquisition of sets of NFID complex data points. Operation of the command is essentially the same as the RA command with DMODE and AMODE having the same affect. Each new shot is not averaged, rather it is displayed and stored starting at points 1 and 4097. Acquisition continues until stopped with the ST command. Execution of this command sets display limits to 1 to NFID.

Defaults: NFID = current value

Range: NFID = 256 - 4096, multiple of 256.

54. ST [STOP ACQUISITION PROCESS]

Format: ST

This command stops acquisition of data begun by RA, CA, SS, or TRRA. Upon execution data buffers are copied to scratch record 1 if acquisition was started by RA or CA.

Defaults: No parameters to default.

55. SWORD [SET START WORD]

Format: SWORD STRWD

This command sets the current default value of the parameter SWORD used by TRRA. STRWD must be between 1 and 8192. It is reset to 1 if TRRA attempts to collect points past 8192 in the data buffer. STRWD = NULL will cause a printout of the current value of TRRA'S SWORD parameter.

Default: STRWD = NULL

56. TEMP [EXECUTE USER DEFINED TEMP PROGRAM]

Format: TEMP PARA1,...,PARA10

This command reads into core the overlay containing the user defined TEMP program and begins execution of that program. Up to ten command line parameters may be accessed by the program to control its operation. The program can make use of all the SPEC disk files and library subroutines and can even execute other SPEC overlays. For details on writing and implementing a TEMP program, refer to the appropriate chapter.

Defaults: Parameter defaults depend on the current TEMP program loaded in SPEC.

NOTE: On first creating SPEC, this command is a no-op and an error message will be output to the console.

57. TI [SET ACQUISITION TIME]

Format: TI ITIME, LNH

This command normally sets the data acquisition dwell time in microseconds. ITIME must be an integer between 3 and 1638. The command loads register D2H with -10*ITIME which is usually used as the sampling rate. The actual use of this register depends on the current pulse program. After loading the register the pulse programmer is initialized by starting execution at line number LN. If LN = 0 this initialization step is skipped. After initialization, the computer waits for a CNTL/F prompt from the pulse programmer.

Defaults: ITIME = 10, LN - A0

Ranges: ITIME = 3 - 1638, LN = 01 - FF

58. TRRA [RUN 2D AVERAGE]

Format: TRRA NTIMES, NTFID, NFID2, ISTART

The TRRA command allows the collection of either a transverse (2D) FID or stacking of data blocks in the data buffer. The incoming data is stored and averaged initially in a temporary data buffer in core. Averaging is in the same manner as the RA command with NTIMES having the same meaning. The DMODE and AMODE commands have the same effect as with any other acquisition command. Averaged data from the temporary buffer is displayed during acquisition. If averaging is stopped by the CU command, that command then operates on the temporary data buffer in the usual manner. All other commands which specify BUF, LOW, HIGH as parameters operate as usual on the specified points of the normal data buffer although the display will not be updated until NTIMES shots are taken or acquisition is stopped by ST. The LIM command, although causing no crash, is non-functional during TRRA. The number of points collected in the temporary buffer during averaging is determined by NTFID. NTFID may take on any value from 1 to 1024.

Once averaging is completed, data is swapped into the normal data buffer. The command then returns to collect more data. The process is continued until enough blocks of data (NTFID long each) have been collected in the data buffer to reach point number NFID2. This means that NTFID must divide NFID2 evenly.

The location of each block swapped to the data buffer is determined by ISTART and the number of the block. As an example, consider the first block. If ISTART = 1, the the block is swapped to the data buffer starting at point 1. If ISTART = 0, then the data starts at point SWORD whose value may be set initially with the SWORD command. After the first block, successive blocks are swapped to consecutive locations in the data buffer. The result is a stacking of data blocks in the data buffer. For the special case where NTFID = 1, this produces a 2D FID. After each swap, the value of SWORD is updated to point to the start of the next block and the updated value is output to the console. If, on its next incrementation, SWORD would become > 8192, it is reset to 1. If SWORD becomes $\geq$ the value of NFID set by the commands NFID, RA, or SS then the contents of the data buffer are written into scratch record 1.

After the collection of each further data block the data will also be written and so NFID should be set equal to NFID2 prior to executing TRRA to cause the data to be written only once.

Defaults: NTIMES = 0, NTFID = current value, NFID2 = current value, ISTART = 0 (start at current sword location).

Ranges: NTIMES = 0-32766, NTFID = 1, 1024, NFID2 = 1, 8192, ISTART = 0,1.

59. WR [WRITE DATA]

Format: WR REC, MASK

This command writes the first 1024 (complex) points of the data buffer into archive record # REC, with specifications set by MASK. Unless inhibited by MASK, the command asks for a title of up to 120 characters (1 1/2 line on ADM3 CRT) which is stored with the data. Also stored are the current acquisition parameters (current shot #, NFID, ITIME), current pulse programmer parameters (loop register values), current Fourier transform parameters (NFFT, scale factor), and mix parameters (PHC, PHL). If the record specified is full then data is written to the next open record. The write pointer is updated. Data will never overwrite a titled record and will only overwrite an untitled full record if specified by MASK. MASK settings are as follows:

BIT 15 must be clear to enable data transfer.
BIT 14 must be clear for title request.
BIT 13 must be clear to print record number.
BIT 12 must be clear to prohibit overwrite of a record with a blank title.

Defaults: REC = record pointed to by write pointer, MASK = 0.

Examples: WR REC,1$ positions pointer to REC without writing.
WR REC,2$ writes to REC without requesting a title.

60. XAVE [CROSS AVERAGE]

Format: XAVE SREC, NX, BUF, LOW, HIGH, LOW2, HIGH2

This command obtains a cross section of averages of a set of records. SREC is the first record averaged, NX consecutive records are done. The value output is the average from LOW to HIGH minus the average from LOW2 to HIGH2 for the buffer(s) specified. The averages are stored in sequence in the data buffer, with SREC's average as point 1, etc. Specifying LOW2 = HIGH2 = NULL omits the second average and subtraction.

Defaults: SREC = current read pointer position, NX = 1,
current BUF, LOW, HIGH; LOW2 = HIGH2 = NULL

NOTE: Every time a data archive is written, the scale factor of the last Fourier transform performed before the writing is recorded with the archive. The XAVE command was designed for cross averaging spectra, and all records averaged across are 'normalized' based on these recorded scale factors. There will be no problems if the data being XAVE'd across were produced by a series of Fourier transforms. If the data were not produced by the FFT command, however, care must be taken to insure that all archives were written with the same FFT scale factor in effect. Otherwise an undesired scaling of some of the data records will result during XAVEing.

IV. SUPPORTING PROGRAMS

In directory SPEC there are a number of supporting programs which help determine the contents of the archives, move records to backup floppys and retrieve them, restore erased records, etc. In the following program descriptions, "Init Fixed Disk" means use the command INIT DP0F:RESERVED, "Release Fixed Disk" means RELEASE DP0F:RESERVED. All programs are executed while in directory SPEC. To get into directory SPEC, type DIR SPEC.

1. ARCHSAVE - restores erased archive records.

In the event that archive records are inadvertantly lost by the ERASE command in SPEC and if they have not been written over with new data, they may be restored with program ARCHSAVE. Titles, however, are permanently lost.

Init Fixed Disk
ARCHSAVE
(Enter low and high record numbers to be restored. Records from low to high, inclusive, are restored.)
Release Fixed Disk

2. CATALOG - reports archive titles.

Archive titles are output to the console by number. Untitled archives produce just the number with a blank title. Empty records produce no output.

Init Fixed Disk
CATALOG
(Enter low and high record numbers. Entering a negative number for each will stop program execution.)
Release Fixed Disk

3. CATLIST - reports archive titles.

Archive titles are output to the console by number and to file DP0:DG.IM. Untitled archives produce a number with a blank title while empty records produce no output at all. The listing which is produced is single spaced--A hard copy may be obtained by printing file DP0:DG.IM.

Init Fixed Disk
CATLIST
(Enter low and high record numbers. Entering a negative number for each will stop program execution.)
Release Fixed Disk

4. DGPP - retrieves pulse programs from a backup file.

Restores PP archive records previously backed up with PPDG. Pulse program, title, and default parameters are restored.

WARNING: This program will overwrite existing PP archives. Make sure that destination record numbers are empty.

DIR DP0
Init Fixed Disk
XFER backup, DG.IM
(backup = backup file, which may be a floppy file)
DIR SPEC
DGPP
(Enter low and high PP archive record numbers. These are the limits of the destination records. If LOW < 0, program execution stops.)
Release Fixed Disk

5. PPDG - backs up pulse programs to backup file.

This program transmits the contents of pulse program archives to the file DP0:DG.IM. These may then be XFER'ed to floppy. Pulse programs, titles, and default parameters are transferred.

WARNING: This program destroys previous contents of DP0:DG.IM.

Init Fixed Disk
PPDG
(Enter low to high archive records to be saved. If LOW < 0, program execution stops. Records from low to high, inclusive, go to DP0:DG.IM.)
DIR DP0
XFER DG.IM, backup
(backup = destination backup file, may be a floppy file.)
Release Fixed Disk

6. PPCATALOG - reports contents of PP archive.

PPCATALOG outputs the contents of the pulse program archive. Titles and, optionally, line listings of the records are output. Output goes either to the TTY or to DP0:DG.IM.

WARNING: Output going to DG.IM will destroy its previos contents.

Init Fixed Disk
PPCATALOG

(Program asks for destination of output. Enter low and high record limits. If LOW < 0, program execution stops. If HIGH = 0, after title output, the program will ask for line number limits. Valid line numbers are from 1 to 64.)

Release Fixed Disk

If output is sent to DP0:DG.IM. form feeds are inserted between program listings. This file may then be sent to the line printer for a hard copy.

7. RARCH - retrieves data archive records from a backup file.

Records previously saved via WARCH are read from a backup file (default is DP0:DG.IM). The backup file may be on a floppy. Archive data, title, and record parameters (FFT SF, etc.) are restored.

WARNING: Previous contents of SPEC archive records are destroyed. Make sure that the destination records are empty (by running CATALOG).

If the backup file is on floppy:

Place floppy in drive

INIT DP1

Now to run RARCH:

Init Fixed Disk

RARCH

(Program asks if default input file, DP0:DG.IM , is to be used. If you answer no, program prompts for input file name. If backup file is on floppy, the directory name is DP1. Next, enter low and high destination archive record numbers. Data is transferred and titles are output to TTY. This is continued until a number < 0 is entered. The program then asks if a new input file is to be transferred. Answering yes will return program to start; answering no halts execution.)

Release Fixed Disk

If a floppy was used:

RELEASE DP1

Remove floppy from drive

8. WARCH - transfers SPEC archive records to a backup file.

Use WARCH to make backups of SPEC archive records. Titles, data and parameters (FFT SF, etc.) are saved. The program outputs to any file, including one on floppy. The default file is DP0:DG.IM,

WARNING: Previous contents of output file, if it exists, are lost.

If output is to go to floppy:

Place a viable (formatted and initialized) floppy in the drive.

If floppy contains no data:

INIT/F DP1

WARNING: This erases previous contents of floppy.

Or if floppy has been used before:

INIT DP1

Now to run WARCH:

Init Fixed Disk

WARCH

(Program asks if default output file, DP0:DG.IM is to be used. If not, program prompts for output file name. If output is to go to floppy, this is DP1:Filename, where Filename is optional. Then, input low and high records to be transferred. Records are transferred and titles are output to the TTY. Empty records are skipped. This continues until a number < 0 is entered for low. Next, program asks if a new output file is to be used. If the answer is yes, program returns to the start. If the answer is no, program execution stops.)

Release Fixed Disk

If a floppy was used:

RELEASE DP1

Remove floppy from the drive.

Note that many of the above programs interact with the file DP0:DG.IM. Because this file may be frequently overwritten, it is advisable not to keep valuable data stored in it. It is notable that the entire PP archive easily fits on one floppy and that each floppy can hold approximately 64 archive records. In addition to the above programs, directory SPEC contains CTOUR, CXFR, CPUT, COSGEN, and SETUP. These programs are used during SPEC creation and are described in Appendix G.

V. USING SPEC'S TEMP COMMAND

SPEC has a command, TEMP, which when entered with parameters, executes a set of subroutines which are defined by the user. This allows for the inclusion of specialized routines which are not included in the SPEC commands and which can be easily changed. The program which is written can obtain parameters from the command line, use any of the SPEC.LB subroutines, call any overlay (with OVRLAY), and access the SPEC fixed disk files. This chapter presents the requirements of the TEMP routines and describes how they are made a part of SPEC.

1. GUIDELINES

The subroutines which make up a TEMP overlay must adhere to the following guidelines.

a. The routines may be in either Fortran or DG's assembly language. The parent routine (the one which calls all others) must be titled "TEMP". If Fortran is used, the parent routine must be SUBROUTINE TEMP with no parameters (arguments). If assembly language is used, there must be an entry point which is Fortran callable titled TEMP.

b. Command line parameters may be obtained from the communications block with SUBROUTINE PGET, the maximum number being 10. Other communications block variable values may be obtained with calls to RCBV and WCBV.

c. Disk records may be accessed with WREC and RREC.

d. The TEMP program must comply with all other SPEC routine characteristics. (See program listings for examples.)

e. The entire TEMP overlay containing all called routines can be no greater than 4096 words of compiled and assembled program steps. An error at RLDR time will tell you if the overlay is too big.

f. Only integer arithmetic operations are allowed. Do not include any floating point numbers.

g. Use labeled common sparingly and avoid DATA statements.

2. STEPS TO MAKE A TEMP OVERLAY

a. Enter all new subroutines to be used with NSPEED (Data General editor).

b. Compile all routines.

c. Create a SPEC TEMP overlay by the following commands (DIR SPEC must be initialized):

```
DIR DP0
RLDR/Z SPEC:OVR27/S SPEC:<SPEC,GLOBL,SPEC.LB> FORT.LB 10000/N^
SPEC:<XFR27,PTEMP> (TEMP ROUTINES) SPEC:SPEC.LB 20000/N
```

where "(TEMP ROUTINES)" means all those routines which you have written which will become part of the TEMP overlay. For the Delta computer, replace FORT.LB by HFORT.LB in the RLDR command. If the loader does not produce an error concerning NMAX then all the routines fit within one overlay. If such an error does appear, then you must shorten the size of the TEMP routines.

d. Transfer the overlay to the fixed disk.

If you did not get an NMAX error from the above commands, you can proceed to move the overlay to the fixed disk. Use the commands:

```
DIR SPEC
INIT DPOF:RESERVED
SETUP
    (Transferring one overlay, overlay number = 27
     size  = 4096
     start = 4096)
RELEASE DPOF:RESERVED
```

3. PRECAUTIONS

Since SPEC runs in the absence of the RDOS monitor, Fortran and runtime errors are not detected and will precipitate a crash. Such errors include integer overflow (unless one of the SPEC routines is used) and an array element out of bounds. For this reason, the routines which are to be included in the TEMP overlay should be thoroughly debugged before being made a part of SPEC.

Be careful with disk access in the TEMP programs. In SPEC it is very easy to overwrite archives, PP records, etc. When in doubt, check the record title flag (first element) before writing. For an example of how this is done, see PWR.FR.

Finally, it should be noted that when SPEC is first created, execution of the TEMP command produces an error that no TEMP routine exists. After a routine is loaded, there is no way, short of clear documentation, to determine what the current TEMP routine does. Unless you know otherwise, always assume that the current TEMP routine is not the one that you want and proceed to load the desired routine by the method above.

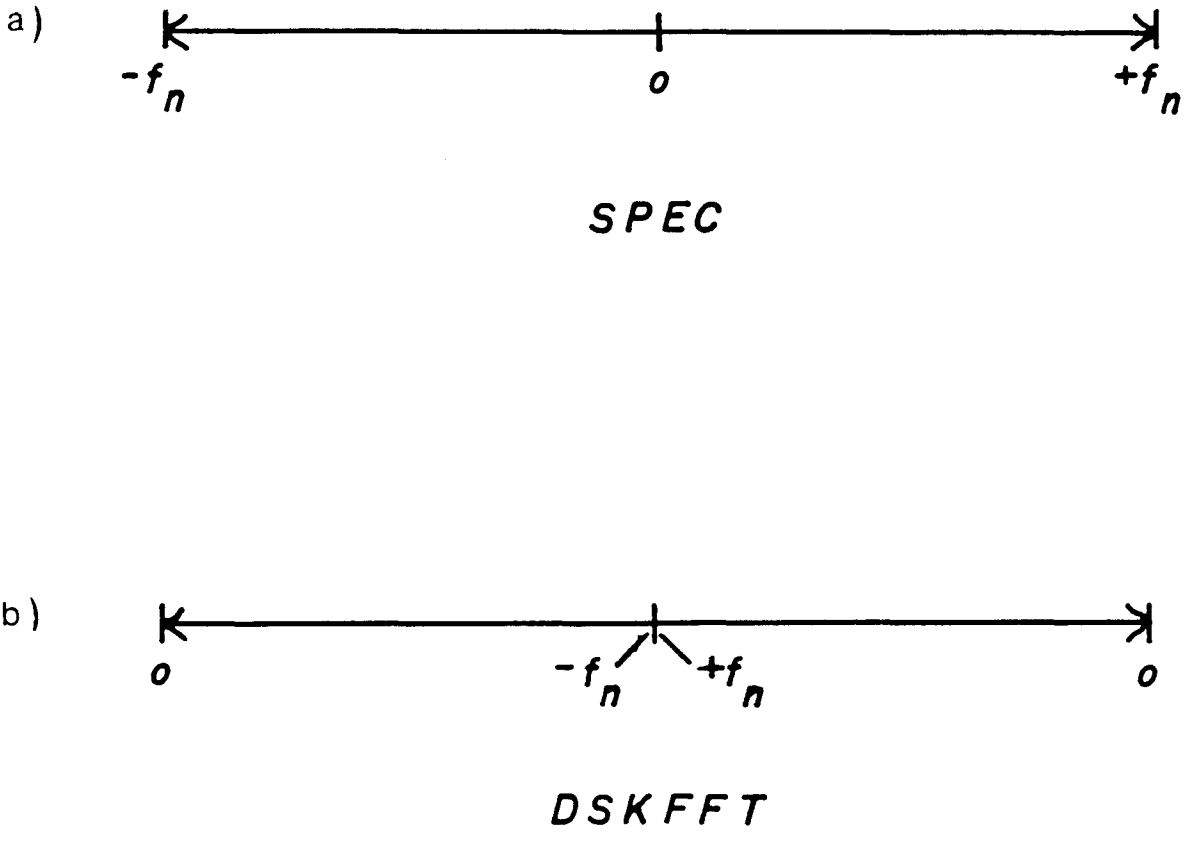
VI. DISK FFT PROGRAMS

SPEC has the capability to do up to an 8K complex (fixed point) Fourier transform. The limitation to this size is primarily memory space. There is a system of programs available which allow one to obtain larger, up to 64K, Fourier transforms. Data may be taken directly out of the SPEC archives or from any backup file, including one on floppy. The programs calculate the Fourier transform (a disk based system relying on a varient of the Cooley-Tukey algorithm), baseline correct data, scale data, calculate its magnitude and add data.

All the calculations are performed on data in the form of floating point numbers. Since data in SPEC is stored as fixed point numbers, it must first be converted to the appropriate form. A program called CONVERT is available for this. In general, this program will have to be run sometime before any of the others. DSKFFT is capable of taking data directly out of the SPEC archive and performing the required conversion. RETSPEC is the program used to return data to SPEC. Floating point numbers are truncated to the interger range, $\pm$32767, before conversion to integer. This is accomplished by scaling floating point numbers so that they will not cause overflow during the conversion.

The frequency scale for the Fourier transform produced by DSKFFT is somewhat different from that in SPEC by FFT. Fig 2 shows the correspondence. Whereas SPEC's scale ranges from the negative Nyquist frequency, through zero, to the positive Nyquist frequency, the scale produced by DSKFFT goes from zero to the negative Nyquist frequency at the center and then from the positive Nyquist frequency back to zero.

In the program descriptions below, parameters within the square brackets are optional arguments to the command line. Some parameters have defaults; others are requested if they are not in the command line. The order of parameters with local switches (file names) is not important. The relative order of numeric parameters is important however. Limits are not specified as LOW and HIGH point limits as in SPEC but, rather, as record limits. Each record is 1K complex points long and requires 8192 bytes of disk storage. The smallest unit which can be specified is 1 record and the greatest number of records is 64. Most of the commands require the starting



XBL 818-1764

FIGURE 2

record number which is between 1 and 800 if it is in reference to the SPEC archives. The parameter BUF in the description below has the same significance as the SPEC parameter. All the programs reside in DIR FFT on the removable pack labeled FFT.

1. CONVERT [INFILE/I, OUTFILE/O, SREC, NREC]

This program converts integer data in either the SPEC archive records or a file created by WARCH to floating point data.

INFILE is the name of the input file (may be floppy). Data is read from sequential file INFILE, without titles, and converted to floating point representation. Result is written to outfile. If INFILE = SPEC, records are read from the SPEC archive and empty records are not skipped.

OUTFILE is the name of the output file for converted data.

SREC is the starting record number to be taken from INFILE.

NREC is the number of records to be taken from INFILE.

Defaults:

INFILE = SPEC
OUTFILE = FID.IM (in dir FFT)
SREC = NULL (requested from console)
NREC = NULL (requested from console)

2. DSKFFT [INFILE/I, OUTFILE/O, SREC, NREC, NFFT]

This program performs the disk based FFT calculation. Input data is in INFILE and output spectrum goes to OUTFILE. No scale factor is output as in SPEC because there is no need to avoid integer overflow. Execution times vary depending on such factors as file medium, etc. For INFILE and OUTFILE both on hard disk and a 32K transform, DSKFFT takes approximately 30 minutes.

INFILE is the name of the input file. This must be a random file created by CONVERT unless INFILE is SPEC. For INFILE = SPEC, data is read directly from SPEC archive and the necessary fixed to floating point conversion performed.

OUTFILE is the name of the output file. If OUTFILE exists, its previous contents are lost. OUTFILE = INFILE will cause problems.

SREC is the starting record number to be taken from INFILE.

NREC is the number of records to be taken from INFILE. (The length of the FID.)

NFFT is the length of the FFT (in 1K units). If NFFT is greater than NREC, zero filling in the manner of SPEC's FFT is done. NREC must be between 8 and 64.

Defaults:

INFILE = SPEC

OUTFILE = FFT.IM

SREC = NULL (requested from console)
NREC = NULL (requested from console)
NFFT = NULL (requested from console)

3. DSKBASE [INFILE/I, OUTFILE/O, BUF, SREC1, NREC1, SREC2, NREC2]

This program performs a (corrected) baseline correction of data in INFILE. The average of NREC2 records from SREC2 is subtracted from NREC1 records starting at SREC1. This is done for the buffers specified by BUF (0, 1, or 2 as in SPEC). Each calculated average is output to the console.

INFILE is the name of the input file.

OUTFILE is the name of the output file.

Specifying only INFILE means that OUTFILE = INFILE.

BUF is the buffer for correction.

SREC1, NREC1 are the starting record and number of records to be corrected and the records to receive the result.

SREC2, NREC2 are the starting record and number of records to be averaged over.

Defaults:

INFILE = FID.IM
OUTFILE = FID.IM
BUF = 0
SREC1 = NREC1 = NULL (requested from console)
SREC2 = SREC1
NREC2 = NREC1

Note that if INFILE = OUTFILE = FID.IM (the default), then the final output file is renamed to BASE.IM.

4. DSKMAG [INFILE/I, OUTFILE/O, SREC, NREC]

Calculates the magnitude of data (same algorithm as SPEC's MAG). Result goes to real buffer.

INFILE is the name of the input file.

OUTFILE is the name of the output file.

If only INFILE is specified then OUTFILE = INFILE and result returns to origin records. Other records left unchanged.

SREC is the number of the first record to be taken from INFILE.

NREC is the number of records to be taken from INFILE.

Defaults:

INFILE = FFT.IM
OUTFILE = FFT.IM
SREC = NREC = NULL (requested from console)

Note that if INFILE = OUTFILE = FFT.IM (the default), then the final output file is renamed to MAG.IM.

5. DSKSCL [INFILE/I, OUTFILE/O, BUF, SREC, NREC, NUM, DENOM]

Scales data by FLOAT(NUM)/ FLOAT(DENOM).

INFILE is the name of the input file.

OUTFILE is the name of the output file.

If only INFILE is specified then OUTFILE = INFILE and result returns to origin records. Other records are left unchanged.

BUF is the buffer to be scaled.

SREC is the starting record to be taken from INFILE.

NREC is the number of records to be taken from INFILE.

NUM, DENOM (integers) specify the scale factor as described above.

Defaults:

INFILE = FFT.IM
OUTFILE = FFT.IM
BUF = 0 (both buffers)
SREC = NREC = NULL (requested from console)
NUM = DENOM = NULL (requested from console)

Note that if INFILE = OUTFILE = FFT.IM then the final output file is renamed to SCALE.IM.

6. DSKBADD [FILENM1/I, FILENM2/[O,R], BUF, SREC1, SREC2, NREC]

Adds complex data in one file to that in another.

FILENM1 is the name of the first file.

FILENM2 is the name of the second file.

If FILENM2 is specified with the O local switch, then result is output to NREC records in this file starting at SREC2. If the R local switch is used, then the result is returned to NREC records in FILENM1 starting at SREC1.

BUF is the buffer to add.

SREC1 is the starting record number in FILENM1.

SREC2 is the starting record number in FILENM2.

NREC is the number of records to add.

Defaults:

FILENM1 = FFT.IM
FILENM2 = BADD.IM
BUF = 0
SREC1 = SREC2 = NREC = NULL (requested from console)

7. RETSPEC

This program is used to return floating point data to SPEC archives as integer data. The program asks questions about the destination archive records, number of records, input record numbers and input file. Input data is first examined to determine the maximum and minimum magnitudes in each of the buffers. The absolute maximum number among these then determines the scale factor by which data will have to be scaled to avoid integer overflow on conversion. Alternately, different max and min values can be chosen and the data is first limited to this range before determining the scale factor. The resulting scale factor is output to the console. Titles may optionally be input to the SPEC archives produced. If the are, they may be up to 80 characters long and are terminated by a carriage return. If there are full titled or untitled archive records in the range specified for output, they are skipped and the next available record is written to. Records numbers written to are output to the console.

Note that the scaled data that RETSPEC produces replaces the original data in the input file. For this reason it is advisable to use a copy of important data.

VII. PULSE PROGRAMMER MANUAL

A. Introduction

The heart of the spectrometer is its microprocessor based pulse programmer. This pulse programmer controls all the gating and timing functions associated with carrying out an experiment and collecting data. The user specifies the desired sequence of pulses and delays through a string of commands which are stored in the memory of the microprocessor. Collectively these commands, which include software looping capabilities, are referred to as the pulse program. At the time of execution, these commands are interpreted in terms of actual gate openings and timing delays by the microcode, which is also stored in the microprocessor. The user is referred to "NMR Studies of Liquid Crystals and Molecules Dissolved in Liquid Crystals", Gary Drobny's Ph.D. thesis, for details of pulse programmer hardware and microcode information. This section is intended simply as a "how to" description for loading and executing pulse programs.

Currently there are two different microcodes which are in use on our spectrometers: The β and δ machines support a microcode written by Jerry Ackerman while the γ is currently running a newer version designed by Gary Drobny. These two microcodes are stored in directory UP as files UPJA.IM and UPGD.IM, respectively. It is important to use a version of SPEC which is specifically tailored for the microcode which you choose to use. Similarities and differences encountered in designing pulse programs for these two different microcodes will be discussed throughout this section.

In using the pulse programmer, the first step is to be sure that the

microcode is properly loaded into the microprocessor. The procedure for doing this is discussed in Appendix E. The appropriate microcode file UPJA.IM or UPGD.IM should be transferred to file UP.IM using the RDOS XFER command before beginning this procedure. The user can tell which version is currently stored as UP.IM by examining the size of this file (2871 = UPJA.IM, 4437 = UPGD.IM).

B. Resetting the Pulse Programmer

The pulse programmer is reset by connecting the microprocessor cable to the main port of the terminal (either directly or through an EIA/TTL toggle box) and hitting the RESET button. The prompt "01." should then appear. For the Ackerman microcode the appropriate response is "14.G", which should elicit a "READY" from the microprocessor. The response for the Drobny code is "10.G" which should bring back the message "UPCODE VERSION 3.255 CMD > ". The microprocessor now awaits further instructions.

The Drobny microcode comes up with the echo turned on. This makes it very convenient to talk directly to the microprocessor and the user is advised to write and debug pulse programs in this manner. Before returning control to SPEC, remember to turn the echo off using the EC command discussed below. The Ackerman microcode comes up with the echo turned off. In order to operate the pulse programmer in stand-alone mode each line of input must be preceded by a CNTL/E and a space and terminated with a CNTL/B. This is not particularly convenient for the editing of pulse programs, and this work is usually done directly from SPEC using the PPTALK command.

C. Command Format

Pulse programmer commands can be divided into two general groups: (1) console commands, which allow the user to input and edit pulse programs,

load pulse programmer registers, and initiate pulse program execution and (2) program commands, which make up the body of the pulse program, and are directly responsible for gate openings, timing delays, software loops, etc. Console commands may be entered directly in response to the "CMD > " or "%" prompts. In the case of the Drobny microcode, the console commands are the only valid responses to the "CMD > " prompt. Program commands are generally entered, with line numbers, for inclusion in the pulse program. When operating under the Drobny microcode, program commands are entered after first executing the console command ED (see below). In the Ackerman version, line-numbered program commands are entered in response to the "%" prompt. In addition many of these commands may be executed directly from the terminal by omitting the line number. This is discussed further below.

D. Console Commands

The commands line format is:

COM OP1 OP2

where COM specifies the command and OP1 and OP2 are operands used in executing the command.

The following console commands are currently available:

1) CL Clears the pulse programmming area.

2) EC VAL

If VAL = 0, echo is turned off. If VAL ≠ 0, all further input is echoed. As noted above, the echo must be turned off for interaction with SPEC, and all command lines must begin with CNTL/B and end with CNTL/E when the echo is off.

3) VA REG

Used to obtain the contents of register REG. The Ackerman microcode's response is the contents of REG in hexadecimal, while the Drobny code gives the value of REG in 1) hexadecimal, 2) positive decimal, and 3) negative decimal.

4) LO REG VAL

Loads register REG with the value VAL. VAL must be a 4-digit hexadecimal number.

5) LI LN1 LN2

Lists the current pulse program from line LN1 thru line LN2. In the Drobny microcode, parameters LN1 and LN2 must always be specified. In the Ackerman code, LI LN1 can be used to list a single line in the program, while LI without parameters list the entire pulse program.

6) GO LN

Initiates pulse program execution at line number LN. Execution continues until a HA statement is encountered within the program. If line LN is unused, an error is returned.

The Drobny microcode has the following two additional console commands:

7) DF REG VAL

Loads register REG with the value VAL. VAL must be a positive or negative decimal number.

8) ED

Causes entry into the pulse programmer editor. The editor will respond with the prompt ED > at which time program commands may be entered. The editor is exited by typing Q(CR).

E. Program Commands

As noted above, many of these commands can be executed directly in response to a "%" prompt in the Ackerman microcode. To do so, leave a space after the "%" and enter the command without a line number. For inclusion in a pulse program, program commands are entered in the following format in response to the "%" prompt (Ackerman) or "ED > " prompt (Drobny):

LN COM OP1 OP2

where LN is a valid line number, COM is a program commands, and OP1 and OP2 are operands used in executing the command. The maximum number of program lines is 64 in the Ackerman microcode and 255 in the Drobny version. Lines need not be entered in numerical order and typing a line number followed by a return deletes that line from the program. A total of 128 16-bit registers are available, numbered 02 to FE (hexadecimal), using even numbers only.

The following programs commands are available; those marked with an asterisk cannot be executed directly from the terminal under the Ackerman microcode. None of the program commands can be executed from the terminal in the Drobny microcode.

1) CO REG1 REG2

Compares contents of register 1 (C(REG1)) to contents of register 2 (C(REG2)). The comparison code COMP is set as follows:

(i) COMP = 01, if C(REG1).EQ.C(REG2)

(ii) COMP = 02, if C(REG1),LT.C(REG2)

(iii) COMP = 04, if C(REG1).GT.C(REG2)

2) *BR LN CODE

Causes a branch to line LN if (CODE.AND.COMP) is non-zero. COMP

refers to the last executed BR statement and branch condition codes are given by:

CODE	BRANCH CONDITION
0(000)	NEVER
1(001)	.EQ.
2(010)	.LT.
3(011)	.LE.
4(100)	.GT.
5(101)	.GE.
6(110)	.NE.
7(111)	ALWAYS

3) DE REG1 REG2

Decrement the contents of REG1 by the contents of REG2, storing the result in REG1.

4) HA

Halts pulse programmer exécution.

5) IN REG1 REG2

Increments the contents of REG1 by the contents of REG2, storing the result in REG1.

6) 01(02,03) REG1 REG2

Output timing word C(REG1) and gate word C(REG2) to the RAM 1(2,3) memory. Currently, the pulse programmer has only RAM 1.

7) OF REG1 REG2

Outputs timing word C(REG1) and gate word C(REG2) to the FIFO.

8) PA NIS

Sets "next instruction source" code on all subsequent OF and 01(02,03) statements.

The source code is : 00 = FIFO

01 = RAM1

02 = RAM2

03 = RAM3

9) R1(R2,R3)

Resets RAM 1(2,3) address counter.

10) *RE

Works only with pulse programmer having START/STOP buttons. If STOP button is pushed, pulse program execution pauses upon encountering RE statement. Commands may be entered from the console until START button is pushed, at which time the program resumes at the command following RE.

The following additional program commands are available with the Drobny microcode. As such, none are directly executable from the console.

11) HN REG1 REG2

Outputs a delay in units of 0.1 μsec to the FIFO. C(REG1) is the length of the delay and C(REG2) is the gate word. C(REG1) must be ≤16384.

12) MS REG1 REG2

Outputs a delay in units of msec to the FIFO. C(REG1) is the length of the delay and C(REG2) is the gate word. C(REG1) must be ≤1000.

13) SC REG1 REG2

Outputs a delay in units of seconds to the FIFO. C(REG1) is the length of the delay and C(REG2) is the gate word. C(REG1) must be ≤255.

14) US REG1 REG2

Outputs a delay in units of μsec to the FIFO. C(REG1) is the length of the delay and C(REG2) is the gate word. C(REG1) must be ≤1000.

15) SB REG1 REG2

C(REG1) is decremented by 1, with the result stored in REG1. If C(REG1) ≠ 0, a branch to line C(REG2) is executed.

16) OD REG1 REG2

Outputs a delay to the FIFO. The length of the delay is given by C(FE) * C(REG1). C(REG1) is the unit time delay, in units of 0.1 μs, and must be ≤ 8192 and C(FE) is a multiplier. C(REG2) is the gate word. Note: If a single OD command occurs in a pulse program, C(REG1) may be any value ≤8192. If, however, two or more OD commands occur, C(REG1) should not be less than about 1500, or timing errors may occur.

17) T1 REG1 REG2

Causes a train of delays to be output to RAM1. C(REG1) is the number of delays in the train and must be ≤255, and C(REG2) is the gate word. C(D2) stores the two's complement of the delay time in units of 0.1 μsec.

F. Gate Words

A total of 20 output ports are available on the back of the pulse programmer. The TTL voltage level at each of these is controlled by a 4-bit hexadecimal gate word. This gate word is specified by C(REG2) in the various output delay commands: O1, OF, HN, MS, SC, US, OD, and T1. Each of the digits in the gate word controls a different group of output ports.

The first digit controls the opening of the four triggering pulse (TP) ports. When specified, each of these TP gates opens for 100 nsec. The manner in which the TP('s) to be opened is (are) selected is illustrated below.

	TP				
	1	2	3	4	
	(2^3)	(2^2)	(2^1)	(2^0)	
TP1, TP3	1	0	1	0	= 9
TP2	0	1	0	0	= 4

The second digit of the 4-digit gate word controls the four auxiliary (AUX) ports. These gates stay open for a time specified by the timing word; the selection of the particular AUX gates is identical to that of the TP's described above.

The third gate word digit specifies which of the A gates is to be opened. These control the r.f. switches on the X-frequency side of the spectrometer. The quadrature phases are controlled by the two least significant bits of the digit. These are coded according to

X = 00 $\quad\quad$ $\bar{X}$ = 10

Y = 01 $\quad\quad$ $\bar{Y}$ = 11

In order to open one of the A gates either AOR1 or AOR2 must also be opened. A sample selection chart for A gates is:

	A				
	1	2			
	(2^3)	(2^2)	(2^1)	$(2^0 2)$	
$A\bar{X}$, AOR1	1	0	1	0	= A
AY, AOR2	0	1	0	1	= 5
AX, AOR1, AOR2	1	1	0	0	= C

The fourth digit in the gate word specifies which of the B gates is to be opened. These control the r.f. switches on the proton side of the spectrometer and selection is identical to that for the A gates. By concatenating the digits for TP's, AUX's, A and B ports together, the 4-bit gate word is generated. Some examples are:

4508 = TP2, AUX2, AUX4, BX

0CB0 = AUX1, AUX2, $A\bar{Y}$

E9A9 = TP1, TP2, TP3, AUX1, AUX4, $A\bar{X}$, BY

Traditionally, several pulse programmer gates have been devoted to specific spectrometer functions. These are listed below:

GATE	FUNCTION
TP1	Data Acquisition Trigger
TP2	Scope trigger
TP4	Phase box increment
AUX1	Receiver de-blank
AUX2	T-controller blank

G. Pulse Programmer Registers

The contents of registers 80H-FEH (even numbers only) are saved along with the pulse program itself when the SPEC command PPSAVE is executed and are restored by PPLOAD. It is therefore important that these registers be used for storing any gate/timing words, parameter values, etc. which are to be saved with the program. The following "universal" registers contain values which are common to all pulse programs. To avoid any possible confusion, the value of the contents of these registers should never be changed.

A0	0	
A2	1	
A4	10	
A6	100	
A8	255	
AA	1000	
AC	127	
C0	-1000	(used in specifying a 100 sec delay)
C2	-2500	(used in specifying a 250 sec delay)
C4	-10000	(used in specifying a 1 msec delay)

Each of the following registers is loaded with the current value of the indicated parameter upon execution of PPLOAD or the designated SPEC command.

REGISTER	PARAMETER	SPEC COMMAND
B0	Loop 1	PPLI
B2	Loop 2	PPLI
B4	Loop 3	PPLI
B6	Loop 4	PPLI
B8	NFID/256	NFID
BA	IRDLY	RDLY
D0	-I90	NINETY
D2	10*(-ITIME)	TI

H. Error Codes

The following is a list of error codes generated by the Ackerman microcode:

C) Error Codes

1) Input line too long

3) a parameter error.

4) b paramater error.

5) Pulse program area filled.

6) Line to be deleted does not exist.

7) Error in hex number.

8) Error in command name.

9) Branch line # error.

10) Attempt to execute a location with no instruction.

11) Overflow in RAM address counter 1.

12) Overflow in RAM address counter 2.

13) Overflow in RAM address counter 3.

14) Attempt to execute unlinked branch.

15) Error in starting line #.

16) Undefined branch line #.

17) Line number error in LI command.

18) Invalid command address.

19) Command not allowable in pulse program.

20) Command may not be performed during recycling.

21) Command may not be executed from console.

I. A POTPOURRI OF SUGGESTIONS TO AID IN THE WRITING, DEBUGGING, AND RUNNING OF PULSE PROGRAMS.

1) The most important thing is to thoroughly debug all pulse programs before connecting the r.f. gates to the high power transmitters. This will insure that no disasters due to seconds of C.W. irradiation occur. In addition, by carefully examining your pulse sequence on the scope, you can often find errors which would prevent your experiment from working properly.

2) In case you encounter any problem while running a program, pressing the RESET button immediately halts program execution and disables all pulse programmer outputs.

3) When operating under Drobny microcode, pulse programs should be debugged while talking to the microprocessor directly. The microcode generates many useful error messages (indicating unlinked branches, parameters out-of-bounds, etc.) which thoroughly confuse the NOVA if you are operating through PPTALK.

4) The first step of every pulse program (including RAM loaders) written for the Drobny microcode should be a CO between any two

valid registers. When this microcode is brought up (following a RESET), COMP is equal to 000 (see CO command described previously). Until the first CO is encountered the unconditional branch statement BR xx 07 will be ignored.

5) It takes 500-600 μsec to reload the FIFO. Be careful not to try to clock this device too quickly. A flashing ERROR light and/or "holes" in an otherwise normal train of pulses are good indications that the FIFO is being emptied in the middle of a pulse sequence. Along these same lines, the first output statement in a pulse program should be a delay of 1 msec or more when using the Ackerman microcode and 20 msec or more when using the Drobny code to allow for initial set-up and stuffing of the FIFO.

6) Be sure to tag the last output RAM (01) statement with a PA 00, specifying return to the FIFO. Otherwise you will loop inside the RAM forever the first time that it is called.

7) Remember to reload the RAM (with a TI from SPEC or GO A0 from PPTALK) each time you change timing/gate words which effect the contents of the RAM.

8) When writing programs for the Drobny microcode, make use of the SB command (which see), keeping the number of discrete CO, BR combinations to an absolute minimum (BR xx 07 is OK). These discrete BR,CO combinations seem to take a long time to execute and program timing errors may result from their use. This explains why many of the programs written for the Ackerman microcode do not run properly under the Drobny version.

9) Try to limit pulse programs written for the Ackerman microcode

to 63 steps. Although a 64 step program does run properly, you will find yourself unable to edit such a program in any way (including attempts to delete a step!)

10) If you leave SPEC for any reason, be sure that register B8H is properly set when you return. Although NFID is automatically set to 1024 upon entering SPEC this does <u>not</u> include a reloading of B8H (If you were running with NFID = 256 and then leave SPEC, you will find NFID = 1024 and B8H = 1 when you return).

APPENDIX

CONTENTS:

A. Bootstrap Procedures

Before anything can be run on one of the computers, it must be "booted". This refers to the process by which the computer is loaded with a disk boot program. This program is the minimum necessary for loading other programs from the disk. It allows one to run formatting and initializing programs, to get into RDOS, and, ultimately to get into SPEC. This appendix will cover bootstrapping procedures for the Nova 2, Nova 820 and Micronova computers. It will be assumed that the disk drive is functional and that there exists a viable removable pack for the hard drives and floppy for the floppy drives. One or the other of these must have previously had the bootstrap installed. If this is not the case, see the appendix on RDOS.

It should be noted that this appendix only attempts to capsulize the information on bootstrap procedures in a convenient location. All of the procedures are fully documented in the DG manuals in lab. Most of these are found in a set of looseleaf notebooks with red binding labels. Although they are marked "MICRONOVA", they contain infomation relevant to all the DG computers. There are several other manuals which fully document the Diablo 4234, the DG 6045 and the floppy drives. It is highly advisable that at least one of these, the small pamphlet entitled "DG DISC DRIVES", be reviewed before any attempt is made to operate one of the units.

1. Nova 2

The Nova 2 system is equipped with a Diablo 10Mbyte hard disk system and a 315Kbyte floppy drive. First, if the drive is not on, turn it on. The "LOAD" and "POWER" lights should be illuminated. If the proper disk pack is already installed, proceed to the bootstrap instructions. Otherwise, open the drive by pulling on the front blue bar until the drive is completely out. Next, release the side latches from the pack. Lift the pack cover off. Be careful not to let the inside of the cover become dirty. Slide the latch on the pack handle to the left and lift the handle until it is vertical. Pull the pack out of the drive, place it in the cover and release the handle. Finally, reverse all these steps to install the right pack. Always have a pack installed in the drive to avoid getting dust in the drive. With the new pack installed and the side latches closed over the pack cover, slide the drive back into the rack. Carefully and firmly push the drive in until you hear the interlocks latch.

Bootstrapping

a) Starting with drive in LOAD:

First, turn the LOAD/RUN switch to RUN. Wait for the "READY" light to come on. Then, step through the following instructions on the front panel of the computer.

1. Toggle STOP, then RESET
2. Load into data switches (octal) then Toggle

000376	EXAMINE
060133	DEPOSIT
000377	DEPOSIT NEXT
000376	START

at this point the computer CPU starts up and "FILENAME?" should appear on the console display. The computer has now been booted. You may now proceed to load RDOS or another system.

b) Starting with drive in RUN:

If, for some reason, the computer crashes in RDOS or SPEC, it must be re-booted to continue. Before you can use the procedure in part a) above, the drive heads must be recalibrated. This is accomplished by:

1. Turning the drive LOAD/RUN switch to LOAD and waiting for the "LOAD" light to come on. Now proceed to a) above.

or:

2. Toggle STOP, then RESET and

Load into data switches (octal) then Toggle

000000	EXAMINE
063033	DEPOSIT
065333	DEPOSIT NEXT
063077	DEPOSIT NEXT
000000	AC0 DEPOSIT
001400	AC1 DEPOSIT
000000	START

Now proceed to a) above.

Either method should produce "FILENAME?".

2. Nova 820

Superficially, the Nova 820's are exactly the same as the Nova 2. However, they are equipped with hardware program load which makes the bootstrap procedure somewhat easier. Both 820's may be bootstrapped with the methods described for the Nova 2 or the following method may be used. The 6045 DG 10Mbyte drives are quite similar to the Diablo drive but the differences will be described here. First, when loading a disk pack, you will notice that there are two side latches on the drive's front panel which must be released before the drive can be pulled out of the rack. Keep these latched when the drive is in the rack. Also, there are no side latches inside the drive to hold down the pack cover.

Bootstrapping

a) Starting with drive in LOAD:

Turn the LOAD/READY switch to READY and wait for the "READY" light to come on. Then, step through the following instructions on the front panel of the computer.

1. Toggle STOP, then RESET
2. Load 100033 (octal) into the data switches and toggle PROGRAM LOAD.

The computer should respond with "FILENAME?" on the console.

b) Starting with drive in READY"

You may recalibrate the drive heads by either method in part b) for the Nova 2. Then perform part a) above.

3. Nova 2, 820 bootstrap from floppy.

You can also bootstrap from a properly prepared floppy on either computer. You must use a floppy on which the bootstrap root has been installed (appendix F) and which has the file BOOT.SV. First, turn the floppy drive power on. Open the drive door and carefully slide a floppy in until it stops. Close the drive door. Turn the floppy drive thumbwheel to 0. For the Diablo 4234 drive, put the drive in "LOAD" (LOAD/RUN switch to LOAD) or turn the drive off. For the DG 6045 drive, turn the drive thumbwheel to 1. Now step through the appropriate bootstrapping procedure above. Finally, make sure the drives are turned to their normal thumbwheel settings when you are done.

4. Micronova (MP200)

Make sure the computer power is on. Turn the floppy drive power on. Place a DOS floppy into the drive and close the door (DOS is the Micronova equivalent of RDOS). With the system console on, type the "break" key. You should get an exclamation point (!). Type 33L (no carriage return). After a while you should see "FILENAME?".

5. Getting into RDOS (DOS)

Once the computer has been booted ("FILENAME?"), you can proceed to get into the RDOS (DOS) operating system. This is accomplished by simply typing a carriage return in response to "FILENAME?". Eventually, RDOS (DOS) will come up.

For RDOS, the dialogue from that point is:

```
FILENAME?  (CR)
PARTITION IN USE - TYPE C TO CONTINUE     type (C)
CONTINUE
NOVA RDOS REV 6.52

DATE (M/D/Y) ? (enter date)
TIME (H:M:S) ? (CR)

R
```

You're now in RDOS CLI.

For the Micronova, everything is as above except that the operating system title is MICRONOVA DOS REV 3.30.

The MP200 has a real time clock, so you may want to enter the correct time although it is not necessary.

B. SPEC Initialization

Each time SPEC is entered from RDOS, certain default parameters are set. In addition, the data buffers and display are initialized. The data buffers will contain the data that was saved on disk the last time SPEC was exited (see "LEAVING SPEC" chapter II). The display will show points 1 to 1024 of the real buffer. The low cursor will be at point 1 and the high cursor at 1024. The display scale factor will be set to 1. The read and write pointers will both be set to record 1. In addition, the following default parameters are set.

1. Acquisition parameters
   a. all loop values are 1.
   b. NFID is 1024.
   c. IRDLY of RDLY command is set to 20.
   d. I90 of NINETY command is set to 50 (5.0 microseconds).
   e. ITIME of TI command is set to 10 (10 microseconds).
   f. ISHOT is set to 1.
   g. The AMODE parameters are set to 1,0,0,0.
   h. The display will be updated with each shot (DMODE parameter = 1).
   i. SWORD for TRRA is set to 1 and NFID2 and NTFID are set to 1024.

2. Fourier transform parameters
   a. NFFT is set to 1024.
   b. The scale factor is set to 1 (2**0).

3. Mixing parameters

   The constant and linear phase shift terms are both set to zero.

C. Spectrometer Peripherals

Each computer system is interfaced to several peripheral devices. Below is a current list of these with a partial description. Each device is more thoroughly documented elsewhere.

1. Nova 2 system
   a. Console: ADM3 CRT TTY
   b. Display: Hewlett-Packard 1300A
   c. Disk drive: Diablo type 4234 10Mbyte dual platter.
   d. Floppy drive: Data General type 6030 315 Kbyte drive (single density).
   e. Pulse programmer: Home-built, microprocessor-based. (8080).
   f. Acquisition: Home-built, 10 bit two's complement with ±5 V. range. 2 microsecond conversion time.
   g. Phase box: Daico narrow band delay line 8 bit phase shifter with home-built controller (centered at 30 MHz)

2. Nova 820 systems
   a. Console: ADM3 CRT TTY or Soroc IQ 120
   b. Display: Hewlett-Packard 1304 A or Tektronix 611.
   c. Disk drive: Data General 6045 10Mbyte dual platter.
   d. Floppy drive: Data General 6030 type 315Kbyte (single density).
   e. Pulse programmer: Home-built, microprocessor-based (Z80).
   f. Acquisition: "BETA" system by Spectrometer Data System, 10 bit two's complement with ± 10 V. range, 1 microsecond conversion time.

      "DELTA" home-built, 10 bit, two's complement with ± 5 V range, 2 microsecond conversion time.
   g. Phase box: Daico narrow band delay line 8 bit phase shifter with home-built controller (centered at 30 MHz).

3. MP200
   a. Console: Datamedia DT80/1 terminal or Soroc IQ120.
   b. Asyncronous interface board currently jumpered to run line printer.

In addition, all computers share a Houston Instruments DP1 Complot continuous X,Y incremental plotter, a Tektronix 4662 plotter, and a Dataroyal 5000 line printer.

D. Floppys/Disks

In this appendix instructions are given for formatting and initializing floppy disks and disk packs. These are the steps required before any disk can be used on the computer. It is advisable that the commercial documentation on disc drives be read prior to operating them. In particular, the small pamphlet entitled "DG DISC DRIVES" is very helpful.

I. Floppys

These instructions are for all the 6030 drives attached to Nova's in the lab. Instructions for the 6038 drive on the MP/200 are given elsewhere.

1. Formatting

Before any floppy can be used it must be formatted. This is accomplished by running a program called CDF. Check that the floppy drive thumbwheel is set to 1 and on the hard disk drive to Ø. First, boot the computer from DPO by one of the methods of appendix A. When you get "FILENAME?" type in CDF/A with a carriage return. The computer will read the program and then halt. Turn the floppy drive power on and insert a new floppy with the write protect hole covered. Place 000501 (octal) in the data switches and toggle START. The ensuing dialogue is shown below with your responses underlined.

CDF dialogue

```
Nova Cartridge/Diskette Formatter Program

Please type the Number of Surfaces
(With a Carriage Return). 1

Please type Unit Number
(With a Carriage Return). 1

Please type (octal) Number of Tracks on the Disk
(With a Carriage Return). 115

Thank you...

TTO Band Rate? = 1200  (9600 on Beta)

Disk Has Been Formatted Only...
```

At this point the computer will have halted and you can remove the floppy. Formatted floppys are marked with a "F" and dated. You may proceed to format another floppy by toggling the START switch after you have loaded the floppy. Note that the Baud Rate question is only asked for the first floppy.

On the unit number question above, be sure that you answer with "1" and that this is the number displayed on the floppy drive front panel thumbwheel switch. A mistake here could result in a different storage unit being inadvertantly erased!

2. Initializing

The next step in preparing a floppy is initializing. Again, put the floppy (write protect hole taped) into the drive and get a "FILENAME?". Type in DKINIT with a carriage return. Then follow the dialogue below.

DKINIT dialogue:

```
DISK INITIALIZER  -  REV 06.50

DISK DRIVE MODEL NUMBER?  6030

6030/6038 DISKETTE DRIVE TYPE

DISK UNIT?  DP1

      (The wrong answer here could erase another disk unit!)

COMMAND?  FULL

COMMAND DESTROYS ...
      .       .
      .       .
      .       .
TYPE...       ...TO ABORT WITHOUT LOSS.

NUMBER OF PATTERNS TO RUN (1-5)? 5

***PATTERN #1 (155555)***
         .       .
         .       .
         .       .
***PATTERN #5 (125252)***

        (This takes about 4.5 min.)
***     ALL PATTERNS RUN    ***

Do you wish to declare any blocks bad that are not already in the bad
block table?  NO

      (Default the next three questions with a carriage return.)

FULL DISK INIT COMPLETE.

COMMAND?
```

At this point you can take out the floppy. Formatted and initialized floppys are marked with "F/I" and the date. If you are done, type STOP to the last question. If not, put in the next floppy, type FULL and proceed as above.
After you are all done, the computer will be halted.

3. Final Initialization

The last necessary step is executed while in RDOS. Get into RDOS by one of the methods in appendix A. With the floppy in its drive, enter the following command.

INIT/F DP1

Make sure you do this right. An error here could erase another storage unit! Now the floppy is an RDOS directory with directory name DP1.

4. Installing Bootstrap (optional)

In order to bootstrap from device DP1 as described in appendix A, you must install the bootstrap root on the floppy. To do this, first get a "FILENAME?" and type BOOT. Then follow the dialogue below.

```
BOOT  REV 6.50

BOOTSTRAP DEVICE SPECIFIER?  DP1

INSTALL BOOTSTRAP (Y or N)? Y
      (no carriage return needed)
DONE.

BOOTSTRAP DEVICE SPECIFIER?  DP0

INSTALL BOOTSTRAP (Y or N)?  N

FILENAME?
```

Finally, while you are in RDOS, execute the following commands

```
DIR DP0
R
INIT DP1
R
MOVE/V  DP1  BOOT.SV

BOOT.SV                  (verification by computer)
R
```

Now you may boot from the floppy by the methods of appendix A.

5. Releasing floppy

Whenever you want to use a floppy in RDOS you must insert the floppy in its drive and type INIT DP1. Before you take a floppy out of the drive, make sure you first release it by typing:

RELEASE DP1

II. Disks

1. Removable Pack

Both the Diablo drive and the DG drives require the same type of disk pack. This is a 5Mbyte pack with 12 hard sectors and may be ordered from DG or BASF. DG packs are more expensive but come already formatted. If an unformatted disk is to be used, it may be formatted with the CDF program in a manner similar to floppys. First, boot the computer and get a "FILENAME?". Then, type in CDF/A and, when the computer halts, put the unformatted pack in the drive. When the drive comes ready, proceed as for a floppy, but answer that there are 2 surfaces and that the (octal) number of tracks is 630.

When the disk is initialized, all the dialogue is the same as that for floppys except where it asks for information about the drive model number and unit. First, get a "FILENAME?" from another disk or floppy and answer DKINIT. Now, put in the pack to be initialized. When the program asks for the model number, answer 4234 for the Diablo drive and 6045 for the DG drives. At the point where it asks for the disk unit, answer DP0. For the DG drives, the thumbwheel must be set at 0. All other dialogue is the same as for floppys.

When the initialization is complete the bootstrap must be installed. The easiest way to do this is to boot from a floppy which itself has a bootstrap root (see under Floppys above). The answer to "FILENAME?" is BOOT this time. Follow the dialogue below.

```
BOOT  REV  6.50

BOOTSTRAP DEVICE SPECIFIER?  DP0
      (drive thumbwheel @ 0 for 6045's)
INSTALL BOOTSTRAP (Y or N)?  Y

DONE.

BOOTSTRAP DEVICE SPECIFIER?  DP1

INSTALL BOOTSTRAP (Y or N)?  N

FILENAME?
```

Now you can use the fixed disk version of RDOS to finish the job. Answer DP0F:FIXED to the above question. When RDOS comes up, do the following:

```
INIT/F DP0      (Do it right!)
R
MOVE/V DP0  BOOT.SV
BOOT.SV
R
```

Now the disk can be booted and is ready to accept any operating system. To do this, see Appendix F.

2. Fixed Disk

The fixed disk may also require formatting and initializing. Only if the disk has crashed and been replaced is it necessary to re-format it. If this (God forbid) is the case, you must simultaneously format a removable pack:

1. First, boot the computer and read in the formatting program from a floppy that has it: FILENAME? CDF/A
2. When asked by CDF, answer that there are 4 surfaces and 630 (octal) tracks.
3. Now initialize the fixed disk by running DKINIT. The answer to the question DISK UNIT? is DP0F. Note that this procedure does not initialize the removable pack.
4. To finish the initialization process, obtain a working RDOS removable pack and install it in the disk drive.
5. Bring up RDOS as usual to receive a prompt:

   ```
   R
   INIT/F DP0F
   ```

6. The fixed disk is now ready to accept an operating system:

   ```
   R
   MOVE/V DP0F SYS.SV, SYS.OL, CLI.SV, CLI.OL, CLI.ER, BOOT.SV
   R
   DIR DP0F
   R
   RENAME SYS.SV FIXED.SV
   ```

```
R
RENAME SYS.OL FIXED.OL
R
BOOT DP0
FILENAME?
```

7. Now remove the RDOS pack and reinsert the one which was formatted along with the fixed pack.
8. Initialize this pack as described in the preceding section and proceed to Appendix F for the rebuilding of RDOS on this pack.

3. Disk Maintenance

There are only a few things to say with regards to disk maintenance. First, keep things clean! Clean the disk heads regularly, according to the manufacturer's specifications. Replace the air filters as required. If the heads or disks are allowed to get dirty, valuable data may be lost and the drive damaged. Finally, keep floppys and disk packs away from the magnets!

E. Reloading Microcode

If, for any reason, the microprocessor on the pulse programmer should crash, its assembled microcode will have to be reloaded. This is also necessary if the pulse programmer is turned off. A symptom that the microcode must be reloaded is if you do not get a READY/CMD > when you initialize it with 14.G/10.G (depending on whether you are operating under the Ackerman or Drobny microcodes (see Sec. VII)). The following steps must be taken to reload the microcode.

I. If you are using the spectrometer's EIA/TTL toggle box:

1. Make sure that the SPEC removable pack is in the disk drive. If so, get into RDOS.
2. R
   DIR UP
   R
   UPLOAD (UPSEND on the Beta Spectrometer)
   (Computer will halt.)
3. Press button on the EIA/TTL toggle box labeled "Micro-P/Main, Nova/Aux".
4. Press RESET on microprocessor. "01." should appear.
5. Toggle CONTINUE on Nova front panel.
6. Microcode loads, last line on CRT should be "00" and Nova will halt.
7. Press RESET on microprocessor. "01." should appear.
8. Type "14.G"/"10.G" (depending on which microcode you are using). This should bring back a response of "READY"/"CMD >".
9. Press button labeled "Micro-P/Aux, Nova/Main" on EIA/TTL box.
10. Toggle CONTINUE on the Nova
    STOP
    R
    (If more should appear, type CNTL/A until you get only the RDOS "R".)
11. Get back to SPEC via
    R
    DIR DP0
    R
    RELEASE UP
    R
    DIR SPEC
    R
    SPEC

12. Press ENABLE (and START) buttons on microprocessor.

II. If you are not using the EIA/TTL toggle box:

1. Make sure that the SPEC removable pack is in the disk drive. If so, get into RDOS.
2. Disconnect the PP cable from the AUX port of the TTY.
3. R
   DIR UP
   R
   UPLOAD (UPSEND on the Beta Spectrometer)
   (Computer will halt.)
4. Replace Nova cable with PP cable on TTY MAIN port (leaving the reversing adapter on the AUX port).
5. Press RESET button on microprocessor. "01." should appear.
6. Place Nova cable on the TTY AUX port.
7. Toggle CONTINUE on Nova front panel.
8. Microcode loads, last line on CRT should be "00" and Nova will halt.
9. Press RESET button on microprocessor. "01." should appear.
10. Type "14.G"/"10.G" (depending on which microcode you are using). This should bring back a response of "READY"/"CMD >".
11. Replace Nova cable on TTY MAIN port and PP cable on TTY AUX port.
12. Toggle CONTINUE on Nova front panel
    STOP
    R
13. Get back to SPEC via
    R
    DIR DP0
    R
    RELEASE UP
    R
    DIR SPEC
    R
    SPEC
14. Press ENABLE (and START) buttons on microprocessor.

F. RDOS/DOS and Data General Computers

This appendix briefly describes some aspects of RDOS and DOS as they are set up on our computers. There is complete documentation of both and the CLI in the Data General manuals. Also discussed here are hardware differences between the computers.

1. RDOS (REV 6.52)

Disk Structure

The RDOS version on the disk drives in lab has several of its utilities in different directories. A list of these directories and a description of their contents is given below. (This list is for DP0.)

Directory	Contents
DP0	Parent directory; CLI, SYS, links to utilities.
UTIL	Assemblers, Lib. File Ed., Text editors, etc.; line printer program; other odds and ends.
F4	All Fortran libraries and Fortran compiler. This directory must be initialized to do any Fortran programming.
SYSGEN	Sysgen program and sysgen dialogue as well as a copy of the system, SYS. This directory must be initialized to run SYSGEN.
RDOSUD652	Update patches for Rev. 6.52 of RDOS.

These basics appear on all of the disks. Things are arranged in this manner for no good reason other than convenience and convention. In order to use utilities such as NSPEED, RLRD, LFE, MAC, FORT, and a few others, directories UTIL and F4 must be initialized. The macro SETUTIL.MC in DP0 does this for you as described below.

In addition to the directories listed above, all disks for use with SPEC should contain directories SPEC (described in Appendix G) and UP, which contains the microprocessor software (see Sec. VII). The disk Fourier transform routines may be found in directory FFT. In order to allow room for large FFT's, it is advisable to put only RDOS directories and directory FFT on the Fourier transform pack. There are various other directories belonging to people for programming use on several of the disk packs.

The fixed disk of each drive contains only one sub-directory, called RESERVED, which contains all of the contiguous data files necessary for SPEC. Since this is so large, there is very little room for anything else on the fixed disk. In the parent directory, DP0F, there are only a few other items. There is a copy of the SPEC program library and initializing program (SPEC.LB

and SPEC.SV). The fixed disk also has its own RDOS system (FIXED) and CLI. DKINIT, CDF, BOOT.SV, and the supporting SPEC programs described in Sec. IV also reside on the fixed disk.

Macros

There are several macro files (.MC extensions) available in RDOS. Below is a list and brief description of each.

Macro Name	Purpose
SENDLPR (in UTIL)	Outputs files to Dataroyal line printer through AUX port of TTY.
LOGOUT (in DP0)	Releases master directory and boots DP0. Execute this before shutting down disk drive.
BACKUP (in DP0)	Backs up current versions on floppys.
SETUTIL (in DP0)	Initializes UTIL and F4.

RDOS Backup Floppys are listed below.

Floppy Title	Contents
DG RDOS DK1 DG RDOS DK2 DG RDOS DK3 DG RDOS DK4 DG RDOS UPDATE	DG release of RDOS Rev. 6.52. Use these to create RDOS from scratch.
DIR DP0	Backup of files and links required in DP0.
DIR UTIL Part 1 DIR UTIL Part 2	Backup of DIR UTIL.
DIR F4	Backup of DIR F4.
DIR SYSGEN	Backup of DIR SYSGEN.
DIR DP0F	Backup of DIR DP0F.

In addition there are 2 copies of SPEC for each spectrometer backed up on floppys.

Recreating RDOS

RDOS may be created on a new removable pack from its backup floppys by the following procedure:

1. Format and initialize the removable pack as described in Appendix D.
2. Create necessary directories:

```
R
DIR DP0
R
CDIR (UTIL, F4, SYSGEN, RDOSUD652)
```

3. Unload the backup floppys to disk directories. For example, for directory F4 do:

```
DIR DP0
R
(Load the F4 backup floppy.)
DIR DP1
R
MOVE/V/A DP0:F4
(Filenames moved are output.)
R
DIR DP0
R
RELEASE DP1
R
```

Do this for each of the directories listed under "Disk Structure" above. Directory RDOSUD652 is backed up as subdirectory UPDBU on the SYSGEN floppy. To unload this directory do the following:

```
DIR DP1
R
DIR UPDBU
R
MOVE/V/A DP0:RDOSUD652
(Release floppy as per above.)
```

Next, make sure that DP0 gets copies of the system (24K and 32K versions) and CLI. These were put into directories SYSGEN and F4 during their creation. To get them to DP0:

```
DIR DP0:SYSGEN
R
MOVE/V DP0 SYS.OL, SYS.SV, SYS32K.OL, SYS32K.SV
R
MOVE/V/D DP0 CLI.OL, CLI.ER, CLI.SV
R
```

You don't have to keep copies of the CLI in directory F4; the CLI files above can be deleted from that directory. If the Nova computer is operating with a full 32K of memory, use the RDOS rename command to make the 32K operating system the default system:

```
R
DIR DP0
R
RENAME SYS.SV SYS24K.SV
R
```

```
RENAME SYS.OL SYS24K.OL
R
RENAME SYS32K.SV SYS.SV
R
RENAME SYS32K.OL SYS.OL
R
```

Finally, boot from your new disk with:

```
DIR DP0
R
BOOT SYS
MASTER DEVICE RELEASED
```

Eventually, RDOS will come up as usual.

The Fortran libraries and macro parameter files were not backed up with the floppys and must be recreated. The general Fortran library, FORT.LB (HFORT.LB for hardware multiply/divide on Delta), is a merged library from several supplied by DG.

To create the FORT.LB, do the following:

```
DIR DP0
SETUTIL (Initializes utility directories.)
LFE M F4:<FORT.LB/0, FORT0.LB, FORT1.LB, FORT2.LB, FORT3.LB, SMPYD.LB>
```

To create HFORT.LB (on Delta) do the following:

```
DIR DP0
SETUTIL
LFE M F4:<HFORT.LB/0, FORT0.LB, FORT1.LB, FORT2.LB, FORT3.LB, HMPYD.LB>
```

If you get error messages from SETUTIL it may mean that directories UTIL and F4 have already been initialized. Proceed with the LFE command line. This loading procedure is described in the DG Fortran IV Users Manual, Appendix D.

In order for the macro assebler, MAC, to operate properly, a file called MAC.PS must exist. This is the "tailored" permanent symbol table (see DG's Macroassembler Users Manual, Chapter 6). You create MAC.PS with the following commands:

```
DIR DP0
MOVE/V/D UTIL RDOS.SR
DIR DP0:UTIL
MAC/S NBID, OSID, RDOS, PARU
```

These commands should only have to be entered once when RDOS is created. There are more details in the DG manuals.

***CONGRATULATIONS...YOU HAVE NOW CREATED A WORKING RDOS PACK...Please label the removable pack as such, including information as to which system (24K or 32K) is the default, which Fortran library (FORT or HFORT) is on the pack, etc. ***

An alternative to this procedure is to build RDOS from scratch from the DG floppys accoring to the method in the DG manual "HOW TO LOAD AND GENERATE YOUR RDOS/DOS SYSTEM." However, this creates an untailored, earlier version of RDOS without the directory structure discussed here. In addition, none of the macros such as "SETUTIL" will be present.

When the commercial RDOS software was received in the form of floppys, the creation of a new system went exactly as described in the "HOW TO LOAD AND..." manual. Unfortunately, problems with the software necessitated acquiring a new copy from DG. Most of the instructions found in that manual are still appropriate. The disks may be initialized as described there. The major difference is that the floppy file FDBOOT.SV does not exist and so the system must be brought up "manually". Follow the instructions for initializing and installing the bootstrap root found in the DG manual and in these appendices. After this, you must use a pack from another computer which has RDOS on it. When you boot RDOS from this pack, initialize DPOF with:

```
INIT/F DPOF
```

Now proceed to copy the diskette to DP0F by:

```
DIR DP0F
INIT DP1
LOAD/V  DP1:1
LOAD    .
        .
        .
```

where the LOAD commands above are from p.2-9 of the DG manual (starting with step #31). All subsequent commands follow that manual.

It is important to carefully read over the System Generation manual before attempting to build RDOS from scratch. You should fully understand what you are doing at each step. You will be using programs and commands which are capable of inadvertantly erasing the DG floppys or another hard disk pack.

Once you've "LOADed" all the utilities from the floppys, you can proceed with the MOVE command to put them on a new DP0. First, put in one of the DG floppys, boot up "FILENAME?" and answer with DP0F:BOOTSYS. Then, with the new DP0 pack in place,

```
INIT/F  DP0
MOVE/V  DP0
    .
    .
    .          (filenames are displayed)
DIR DP0
SYSGEN
```

You may now proceed with the sysgen dialogue as described in the DG manual. SYSGEN creates the tailored RDOS system. A copy of the sysgen dialogue should always be kept in the System Generation notebook.

This procedure builds the RDOS system from scratch and yields a system without any of the macros or directories described in this appendix. For this reason, it is always easier in the long run to remake RDOS from an existing DP0 pack or from the backup floppys as described previously.

Finally, the new RDOS should be backed upon floppys as follows. Six formatted and initialized floppys are required.

a) First floppy; DP0 backup:

```
INIT/F  DP1
DIR DP0
MOVE/ V/A DP1  <SYS.OL, SYS.SV, CLI.->/N,-.FR/N
         files are moved
RELEASE DP1
Mark this as the DP0 backup.
```

If this does not all fit on one floppy, the remaining files may be put on another floppy.

b) Second floppy; UTIL #1 (if it exists):

place new floppy in drive

```
INIT/F  DP1
DIR UTIL
BUILD  U.NM, -.-
```

Use the editor to divide the filenames in U.NM into approximately halves. MOVE one half of the files in UTIL to this floppy.

```
RELEASE DP1
```
Mark this floppy as UTIL Part #1.

c) Third floppy; UTIL #2

MOVE the second half of the files to a floppy marked UTIL Part #2.

d) Fourth floppy; F4

```
DIR F4
```
put in a new floppy
```
INIT/F  DP1
MOVE/V/A  DP1
RELEASE (DP1, F4)
```
Mark this floppy F4.

e) Fifth floppy; SYSGEN

```
DIR SYSGEN
INIT/F  DP1   (new floppy)
MOVE/V/A  DP1
DIR  DP0
RELEASE SYSGEN
MOVE/V  DP1  SYS.OL, SYS.SV, CLI.-

DIR  DP1
CDIR UPDBU
DIR DP0:RDOSUD652
MOVE/V/A  DP1:UPDBU

RELEASE  (RDOSUD652, DP1)
```
Mark this floppy SYSGEN/UPDATE.

f) Sixth floppy: DP0F

```
DIR DP0F
```
put in a new floppy
```
INIT/F  DP1
MOVE/V/A  DP1
RELEASE  DP1
DIR DP0
RELEASE  DP0F
```
Mark this floppy DP0F.

This completes backing up the tailored RDOS with all file links and macro files. If you are all done then release the disk drive before turning it off. This is done by any of:

```
RELEASE DP0  or
RELEASE %MDIR%  or
LOGOUT  or
BOOT DP0  or
BOOT %MDIR%
```

2. DOS

DOS is the floppy based operating system for the Micronova (MP/200). Formatting floppies for use on the MP/200 requires use of the Data General supplied DTOS floppy. Place this floppy in DP0 and execute the normal boot sequence for the MP/200, i.e. toggle the RESET button. The ensuing dialogue is listed below, with the user's responses underlined:

```
077452
!33L TESTOK  (This takes 15-30 seconds.)
```

(Note that a carriage return is not required. Carriage returns will be required after all other user responses.)

```
ENTER CPU SUBTYPE AND CR
(0=601, 1=602 OR MP/100, 3=MBC/1)  1

TOP OF MEMORY =          076777
HIDTOS REV 4.0
DISK ID NUMBER:  1
*LOAD FPY FRMT  (This command takes several seconds to
                 execute.)
LOAD:
FPY FRMT REV.02 FLOPPY FORMATTER
ENTER FLOPPY DEVICE CODE  33
```

(At this point you may remove the DTOS floppy from DP0 and place the blank floppy into either DP0 or DP1.)

```
UNIT #  0  (0 if floppy is in DP0; 1 if in DP1)
NORMAL OR FAST FORMATTING  N
```

You may get the message "DISKETTE NOT READY". If so, open the drive door and pull the floppy, then reload. Sooner or later you will get it to type:

```
LOAD DISKETT HIT CR
```

A carriage return sets the drive whirring away. Formatting takes 2-3 minutes. If you get only the error message, then the diskette is probably bad. Try another. Finally the computer responds:

```
FORMATTING COMPLETE.  MORE?
```

Responding N exits the program, while responding Y returns the prompt:

ENTER FLOPPY ...

Remove the formatted diskette. Initializing is performed with program DOSINIT found on many working DOS floppys and is executed precisely as described in the DG DOS manuals.

***WARNING: DOS is incapable of dealing with bad blocks on a diskette. Should DOSINIT find bad blocks when doing a FULL initialization, the floppy in question should be reformatted for RDOS use. Similarly, when running in DOS the message "SOFT BLOCK ERROR DETECTED ON DPN" suggests that the floppy in drive DPN is going bad and should be copied immediately onto another working DOS floppy before the files become inaccessible. ***

3. Hardware Differences: Novas

The Nova 2 and the Delta Nova 820 both have TTY I/O interfaces running at 1200 baud. The interface on the Beta Nova 820 operates at 9600 baud. The Delta 820 is the only computer with hardware multiply/divide. SPEC on the Delta uses this feature and it speeds up some operations by as much as 30%. Any Fortran program may also use this feature by loading the library HFORT.LB at RLDR time. The decrease in computation time depends on the number of integer assembly language multiply/divides the program does.

The scope interfaces of the three computers are slightly different and this is reflected in the SPEC routine SCOPE. Also, the acquisition system of the Beta system is significantly different than the others, although this causes no difference in its functioning. The Beta's acquisition system places the current data in the 10 least significant bits. This is shifted to the most significant when the averaging is performed so the result is the same as on the other two computers. All the disk drives, floppy and hard, are capable of swapping disks. The two 820's are equipped with hardware program load whereas the Nova 2 is not.

G. Technical Manual for SPEC

This appendix documents how SPEC runs and how to make changes in the program or recreate it from backup.

1. Memory partitioning

When SPEC is first run, certain software partitions are set up which define where in memory data and programs go. These partitititions are defined as entry points in the routine GLOBL and may be accessed when defined as "external" entry points in any routine. Fig.3 shows how the memory is actually partitioned by GLOBL. The bottom 4K of memory is reserved for what is called the root segment. The root segment holds all the routines that are globally required by SPEC's commands. Also contained in this area is the communications block which holds all the variables required to be passed between overlays. (This is a sort of "Labeled Common" area.) Below is a list of all the routines contained in the root segment.

Root Segment Modules from SPEC. These are loaded into memory from from location 1000 (octal) to 100000.

SPEC
GLOBL (includes all software partition definitions)
INIT
BOOT
WCBV TCBN
RCBV
OVRLAY TOVR
WREC LOOKUP DIREC WBLK
RREC
WSTR CGET WTTO

The rest of the 4K root segment is used for the display buffer common area (512 words) and the Fortran library routines. Which Fortran routines are loaded can be determined by examining the SPEC load map created by LINK.CM.

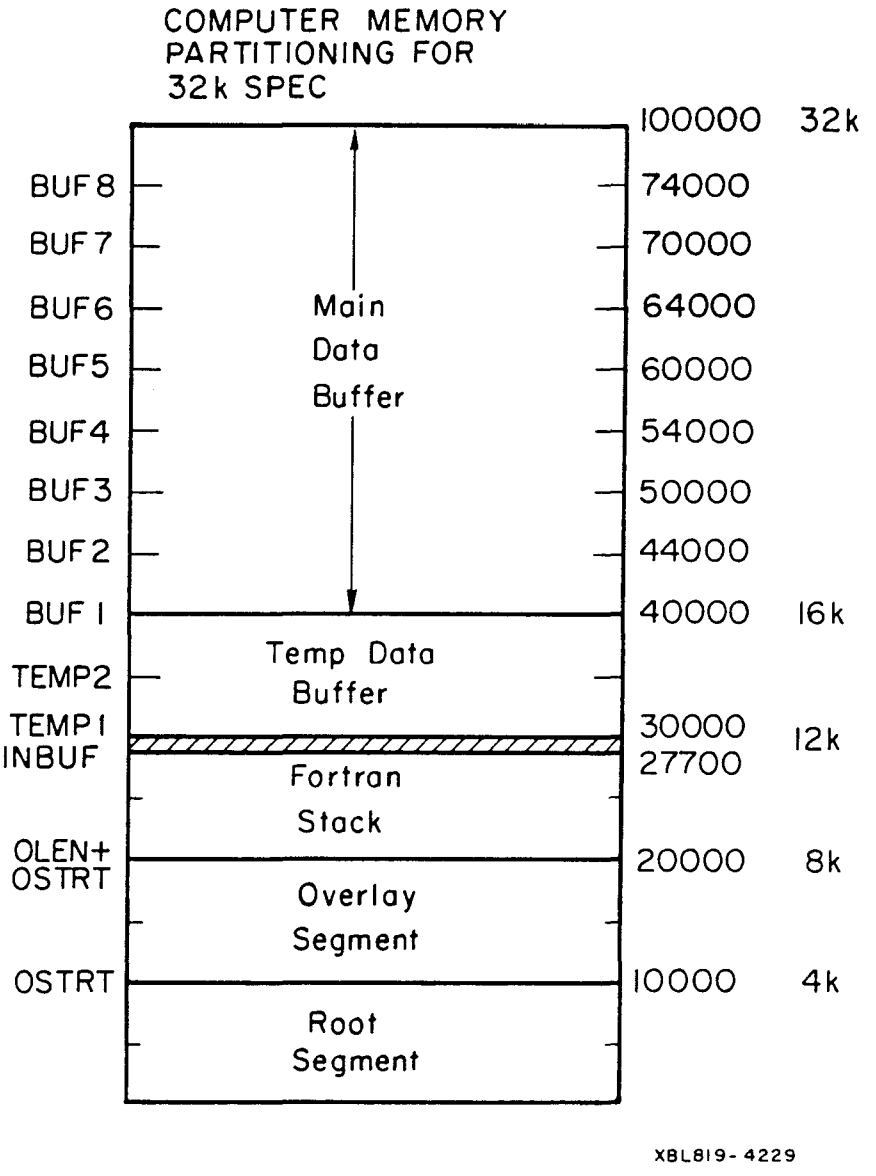

COMPUTER MEMORY
PARTITIONING FOR
32k SPEC
100000
32k
BUF 8
74000
BUF 7
70000
BUF 6
64000
Main
Data
Buffer
BUF 5
60000
BUF 4
54000
BUF 3
50000
BUF 2
44000
BUF 1
40000
16k
Temp Data
Buffer
TEMP2
TEMP1
30000
12k
INBUF
27700
Fortran
Stack
OLEN+
OSTRT
20000
8k
Overlay
Segment
OSTRT
10000
4k
Root
Segment
XBL819-4229

FIGURE 3

In the next 4K of memory above the root segment is the overlay segment. One overlay exactly fits in this space as assured at load time. Each overlay (there are 27 total) may contain one or several routines for SPEC commands. Each time a command is entered (from the console or from a macro) a decision is made as to whether an overlay must be loaded. If the routines required by the command are already in core, a jump to the appropriate entry point is made. Otherwise, the overlay with the appropriate routines is loaded into the overlay segment from disk. The routine OVRLAY handles overlay swapping.

Above the overlay segment is a free area which is used as the fortran runtime stack. The start of this region is set when SPEC is initialized. This must be done because there is no RDOS monitor present to allocate variable space during runtime. The lack of a monitor points out one limitation of SPEC. Any routines which cause a runtime error (such as integer overflow or array element out of bounds) will cause SPEC to crash. This means that all routines must be thoroughly debugged before being incorporated in SPEC.

Memory partitioning varies between the computers at this point. The next partition for the Delta and Gamma SPECs is TEMP1 described below. For the Beta version of SPEC this next partition is INBUF which is 1024 words below TEMP1. The purpose of this region is to provide a depository for incoming ASCII data read from the microprocessor by the routine RSPP. This is required in the Beta because of the higher I/O Baud rate between computer and microprocessor. At 1200 Baud (Gamma and Delta) RSPP reads input from the TTY and checks each character for special significance. At 9600 Baud (Beta) this takes too long and so some data would be lost. RSPP then just collects the input in the computer's memory starting at INBUF. After all data is input, the special characters are detected and removed. The size of INBUF is dictated by the length of a 64 line pulse program listing. The rest of memory partitioning for a 32K version of SPEC follows the same layout as that for the Gamma and Delta.

The next 4K of memory is devoted to the "TEMP" data buffer used by TRRA. There are two 2K regions starting at locations TEMP1 and TEMP2. When TRRA averages, each single shot is stored starting at TEMP2 and the average starts at TEMP1. Location TEMP1 is also given the name BUFFER.

The top 16K of memory is the data buffer. Like the TEMP buffer, this is partitioned into 2K blocks starting at BUF1, BUF2 and so on up to BUF8. All the data acquisition and manipulation routines use this area of memory.

These memory partitions are all defined in the routine GLOBL (file GLOBL.SR). Just defining the quantities themselves does not tell SPEC routines the locations of these software partitions. In addition to the numerical definitions, the partitions are defined as "ENTRY POINTS" by a "ENT" statement in GLOBL. Any subroutine which wishes to use these locations must then access them by using an EXTERNAL statement. For example, in PMAG, the statement

EXTERNAL BUF1

appears. Location BUF1 is then passed to the manipulation routine MAG where it is used as an array. This method is in effect the same as dimensioning very large arrays in Fortran and passing their names as arguments to subroutines. The difference is that now the "arrays" are given permanent absolute starting addresses in memory rather than relying on allocation from a monitor. GLOBL also forces routines required in the ROOT segment to be loaded there.

2. Disk Files

SPEC uses several contiguous files on the fixed disk to carry out its operations. These are contiguous so that the RDOS monitor is not required for random I/O. They reside in DP0F:RESERVED and are created by RDOS at SPEC creation time. SPEC knows of their location via its software (see below). What follows is a list of their names, sizes, current starting locations and contents:

FILE	SIZE in bytes	in disk blocks	STARTING LOCATION Beta	Gamma	Delta	CONTENTS
OVRLAY	221184	432	16263	222	17205	overlays
SCRATCH	131072	256	17454	1102	20375	scratch records
TEMP	32768	64	20055	1503	00224	temporary record
MACRO	51200	100	20156	1604	20775	macros
PPROG	102400	200	17144	1751	20065	pulse prog. archives
PHASE	32768	64	20322	2261	21141	phase box archives
TITLE	409600	800	14622	2361	15545	data archive titles
DATA	3276800	6400	222	4022	01145	data archives
TOTAL:	4257792	8316				

There are two routines which handle SPEC's I/O with the disk. These are RREC, for reading, and WREC, for writing. Both routines call LOOKUP to retrieve necessary information about a particular file. This information is stored in the routine DIREC. This routine must be edited if SPEC is created from scratch. Finally, two entry points, RBLK and WBLK, within IODISK handle the actual disk drive operations.

3. Command Parameter Routines

Whenever a command is executed by SPEC, program flow first goes to one of the appropriate parameter check routines. These are all Fortran subroutines whose titles are the command name prefixed by a P. For example PADD is the parameter routine for the ADD command. These routines check the validity of the parameters entered with the command and, if they are within the proper range, call up the routine that actually does the work. The next paragraph outlines the general program flow of any of these routines.

First, default parameter values are set. Next, the actual parameters entered are retrieved from the communications block within the root segment. PGET does this; RCBV is used to read from the communications block and WCBV writes to it. The resulting parameter values are then checked for errors. If no errors are found, routines are called to execute the command. If any error is found a message is output to the terminal and flow returns to the calling program (OVRLAY).

4. TTY I/O

Since SPEC operates in the absence of the RDOS monitor, it requires its own routines for TTY I/O. WSTR is the routine which is used to write any string out to the terminal. Exclamation point characters are transformed to a carriage return/line feed. RSTR is the routine to read input from the TTY. RSTR recognizes all special control codes. The TTY drivers which test, read and write single characters are TTTI, RTTI, and WTTO respectively. In addition, there are two special routines which send special characters to the TTY for the PP. These are RSPP and WSPP for reading and writing. String data delimited by parentheses in a call to WSPP causes the proper control characters for the microprocessor to also be sent. Likewise, RSPP reads only characters output by the PP (without an echo). As mentioned before, the Beta version of SPEC acquires data from the

TTY in a slightly different way. Input is read (without an echo) by RDNE and placed in a holding buffer between INBUF and TEMP1 in memory until a CNTL/F is detected. RSPP then returns and strips off unnecessary control characters. This method is necessary due to the higher Baud rate of the Beta's I/O interface.

5. Interrupts

In order to service several different devices on a random basis, SPEC uses a routine which simulates an interrupt handler. This is subroutine WAIT. WAIT essentially just loops, displaying the display buffer, while it waits for an interrupt. WAIT can be told to recognize interrupts from either the TTY, the PP or the HSA (high speed acquisiton), or a combination of any of the three. When interrupted, WAIT returns a control code identifying the device that first interrupted it. For the TTY, WAIT is interrupted when any character is struck on the console. For the PP, WAIT is interrupted when a prompt (CNTL/F) is output by the PP. For the HSA, WAIT is interrupted when the total number of points to be acquired have been placed in the Nova's memory. WAIT is used extensively throughout SPEC when it is expected that one of these devices should be able to cause an interrupt.

6. Program flow

The flow of operation whenever SPEC reads and executes a command is as follows. First, when SPEC is run from RDOS, after the program is initialized, the first overlay is loaded from the disk. The routine CMDPRS first receives control. This calls RCOM which decides whether the next command is to be read from the TTY or from the disk (a macro). RCOM returns the command to CMDPRS. PARSE is then used to break the command line down into the command name and its parameters. NSM converts numeric string input into its integer equivalent. CMDPRS writes the converted parameters to the communications block and, if the command name is valid, calls OVRLAY to load the overlay with the command's parameter routine. At the completion of the command execution, program flow returns to OVRLAY which reloads CMDPRS which in turn accepts the next command. A set of internal flags are maintained which enable SPEC to distinguish between input, execution, and acquisition modes.

7. Integer Arithmetic Routines

SPEC does not contain a floating point interpreter, so all arithmetic operations are carried out on integers. Thus, the dynamic range of SPEC is ±32767. However, to make some calculations possible and to enable overflow detection, SPEC contains several double precision arithmetic routines. SSSUM is a single precision addition routine, DSSUM is its double precision version. DSDVD divides a double precision integer by a single precision integer and returns the quotient and a remainder. DDSUB subtracts two double precision integers. DFRACT divides two double precision integers to give a single precision result. The result is +32768 times the quotient. MULDVD multiplies a number by a fraction and handles most of the multiplications and divisions. In this routine, the software or hardware double precision multiplications and divisions are used. For the Nova 2 and the Beta 820, the software routines are used. For the Delta 820, the hardware mult/div unit is used. Finally, ICABS calculates the magnitude (SQRT of sum of squares) of a pair of numbers. This routine is used by the MAG command.

8. Recreating SPEC from Scratch

In the event that SPEC must be recreated for any reason (eg. disk crash), the following steps must be taken. In the following, only the RDOS commands required are listed; any dialogue is only briefly described.

First, obtain the proper backup floppy. All the SPEC source files fit on one floppy. Retrieve backup files by starting in DIR DP0 and executing the following.

```
CDIR SPEC  (if DIR SPEC does not already exist).
INIT SPEC
DIR DP1  (floppy in drive)
MOVE/V DP0:SPEC
```

Next, if the fixed disk has been replaced or erased, recreate RDOS directory DP0F as described in appendix F. Now, make the SPEC fixed disk files by doing:

```
DIR DP0F
CDIR RESERVED (if RESERVED does not already exist)
DIR RESERVED
CCONT DATA 6400
CCONT TITLE 800
```

```
CCONT OVRLAY 432
CCONT PPROG 200
CCONT SCRATCH 256
CCONT TEMP 64
CCONT MACRO 100
CCONT PHASE 64
```

(These should be done in this order or this may not work.)

```
LIST/E
```

(This command produces the starting block numbers for the contiguous files above. They appear in square brackets.)

```
DIR DP0
RELEASE DP0F
```

The starting block numbers from the LIST command must be entered in the file DIREC.SR in SPEC. These numbers are the S- variables in that routine. SPEC must have these numbers if it is to find the fixed disk files. Use the editor to place these numbers in DIREC.SR.

Next, the cosine table must be created. First run COSGEN:

```
FORT SPEC:COSGEN
RLDR SPEC:COSGEN, FORT.LB
DIR SPEC
COSGEN
(Size of the cosine table = 2048.)
```

This produces COSTBL.IM, a sixteen bit cosine table from 0 to 90 degrees. This must be used to produce TCOS.SR, the source listing for the cosine table. To do this, build the source file by

```
APPEND TCOS.SR, TCOS1.TX, COSTBL.IM, TCOS2.TX
```

Now the XFR source files must be created. These are assembly language routines used to load the proper routines from SPEC.LB for each overlay. To create them do

```
DIR DP0
FORT SPEC:(CXFR,CPUT)
RLDR SPEC:<CXFR,CPUT>,FORT.LB
DIR SPEC
CXFR
(number of overlays = 27)
```

Finally, SPEC is compiled, loaded and placed on the fixed disk by

```
DIR DPO
@SPEC:UNBACK.CM@
```

This will ask several questions; the answers are as follows:

```
All or one overlay? (you want all overlays transferred
                     to fixed disk,number of overlays = 27)
SIZE? 4096
START? 4096
```

Error listings for the compilations are found in the files

```
DPO:FRTER.LS
DPO:MACER1.LS
DPO:MACER2.LS
```

Examine these to determine if all compilations went without a hitch. The SPEC load map is found in the file

```
DPO:SPEC.LM
```

Examining this will tell you if all overlays have been loaded properly. After examination, the .LS files and SPEC.LM may all be deleted.

The following is a list of several useful RDOS commands.

a) To compile all SPEC Fortran programs:

```
FORT SPEC:(@SPEC:FORT.NM@)
```

b) To assemble all Macro programs:

```
MAC SPEC:(@SPEC:MAC.NM@)
```

c) To assemble all XFR programs:

```
MAC SPEC:(@SPEC:XFR.NM@)
```

d) To create SPEC.LB:

```
LFE N SPEC:SPEC.LB/O SPEC:<@SPEC:LIB.NM@>
```

e) To link SPEC program:

```
@SPEC:LINK.CM@
```

f) To create TOVR.SR:

```
DIR SPEC
CTOVR
(number of overlays = 27)
```

g) To create XFR source files:

```
DIR SPEC
CXFR
(number of overlays = 27)
```

h) To transfer SPEC overlays to fixed disk:

```
INIT DP0F:RESERVED
DIR SPEC
SETUP
(all overlays,
 number of overlays = 27,
 size = 4096,
 start = 4096,)
```

Note: The system generation programs of SPEC are informed of its overlay structure by reading a file called SPEC:OVR. This file consists of groups of subroutine names seperated by blank lines. Each subroutine group (and the subroutines they call) will form an overlay. After linking, the overlays are named OVR01.SV, OVR02.SV,... OVRnn.SV where the counting is done in decimal. SPEC:OVR is used by CTOVR and CXFR.

9. Making modifications to SPEC

Three cases will be considered in order of increasing complexity.

I. A modification is made to one of the overlays. The changed subroutines must be entered and compiled. If a new subroutine is to be used, the name of its relocatable binary file must be included in LIB.NM. Use the editor to do this. (LIB.NM is a file listing the names of all routines which comprise the library SPEC.LB.) In LIB.NM, file names with -.RB extensions are ordered such that any routine follows all routines which call it. Aside from this, routines are placed in alphabetical order. To accomplish a modification of this type do the following.

```
edit and compile affected subroutine(s)
edit LIB.NM (if required)
DIR DP0
LFE N SPEC:SPEC.LB/O SPEC:<@SPEC:LIB.NM@>
RLDR/Z SPEC:OVRnn/S SPEC:<SPEC,GLOBL,SPEC.LB> FORT.LB 10000/N
  SPEC:<XFRnn,SPEC.LB> 20000/N
DIR SPEC
INIT DP0F:RESERVED
SETUP
RELEASE DP0F:RESERVED
```

where nn is the number of the affected overlay. SETUP will ask questions concerning the number of the overlay to be transferred (nn),

its size (4096) and starting location (4096). For the Delta computer, use the hardware multiply/divide library, HFORT.LB, instead of FORT.LB in the RLDR command.

II. A modification is made to one of the ROOT segment routines. In this case, all overlays must be recreated.

```
edit, compile affected subroutine(s), edit LIB.NM if
necessary
  @SPEC:CREATE.CM@
```

SETUP again asks questions. This time you want it to transfer all overlays.

III. The basic overlay structure is changed. Either the ordering, contents or number of overlays is changed. SPEC may be entirely reloaded or just the affected overlays reloaded. Which is more applicable depends on the extent and nature of the change and the relative time for automatic reloading of all overlays versus manual reloading of only a few overlays. In any case, reloading all overlays is always more general and, while taking more time, requires less human effort.

In this case, the file SPEC:OVR must be edited to reflect the new overlay structure. TOVR.SR and the XFR source files must be recreated. If library subroutines are changed or added, LIB.NM will have to be edited and SPEC.LB recreated. If the number of overlays changes, that number will have to be entered into DIREC.SR (the "NOVR" variable must be changed). The following procedure is then used to recreate SPEC.

First, if the # of overlays has changed, the disk file DPOF:RESERVED:OVRLAY must be remade with the CCONT command as during SPEC's initial creation. The new starting block must be entered into DIREC.SR ("SOVR" variable).

```
edit, compile affected subroutines
edit SPEC:LIB.NM (if necessary)
edit SPEC:DIREC.SR (if # of overlays has changed)
edit SPEC:OVR

DIR SPEC
CTOVR
DIR DPO
```

```
MAC SPEC:DIREC
MAC SPEC:TOVR
DIR SPEC
CXFR (enter new # of overlays)
BUILD XFR.NM, XFR-.SR
DIR DP0
MAC SPEC:(@SPEC:XFR.NM@)
@SPEC:CREATE.CM@
```

answer SETUP with new # of overlays

size and start still 4096

Document new number of overlays in this manual.

There are various other changes which can be made to SPEC. For example, the size or number of the fixed disk files may be changed or the absolute locations of the software memory partitions may be redefined (see GLOBL). Generally, however, such changes will require that SPEC be recreated from scratch. It would be difficult to document a general procedure for implementing such changes. Any attempts to make changes not described by the three cases above would require a thorough understanding of how SPEC operates. Extreme care must be taken to ensure that all affected aspects of SPEC are recompiled and loaded and that there are no execution errors capable of precipitating a "crash". The most direct way of understanding how SPEC operates is to examine the source code listings.

Finally, it is important to create new backups of a "new" SPEC. Do not overwrite previous backup copies-- you may find you need them for later use.

10. Backing up SPEC

All the source and text files for SPEC in its current status fit on one floppy. To place these on a floppy, format and initialize it and then do the following.

```
DIR DP0
INIT/F DP1
DIR SPEC
MOVE/V DP1,-.FR,-.SR,-.CM,-.TX,LIB.NM,OVR,^
TCOS.SR/N,COM.CM/N,XFR-.SR/N
```

This RDOS command will copy all the Fortran and assembly language files to floppy (except TCOS.SR and the XFR files). Also copied are the command files for building SPEC, the TCOS text files, OVR and LIB.NM. The copying process will take several minutes.

H. Printing a File

The following procedure may be used to print a RDOS file from any of the NOVA computers. The lab currently has in it a Dataroyal 5000 line printer. Printing of files from the NOVA's is through the TTY. The routine for printing sends text to the printer one line at a time, with a suitable software generated delay between lines to insure that the printer can keep up.

The macro command file for printing, SENDLPR.MC, is located in directory UTIL and is linked to DP0. The dialogue for printing a file is as follows:

```
R
DIR DP0
R
SETUTIL
R
SENDLPR
PRINTING FILES FROM DIR DP0
COLUMN WIDTH OF PAGE (80 OR 132)?  80
ENTER FILE SPECIFIER:

DG.IM, SPEC:RARCH.FR,....
STOP
SET UP LINE PRINTER
STRIKE ANY KEY TO CONTINUE
```

(At this point you must set up the line printer. Connect the printer to the AUX port of the terminal (no reversing adapter). Make sure baud rate is set properly (1200 baud = 4, 9600 baud = 1), adjust paper to top of page, press RESET (beep) and then ON LINE...Now strike any key. Text will be printed and echoed on TTY.)

```
DISCONNECT LINE PRINTER      (Do it!)
STRIKE ANY KEY TO CONTINUE   (Do it!)
R
RELEASE (UTIL, F4)
```

When a file is printed, its name and the data entered when RDOS was brought up are printed at the top of the page. This is another good reason for entering the correct date when entering RDOS.

TIMING DIAGRAM FOR PULSE SEQUENCE AND ACQUISITION CYCLE